Information Builders

WebFOCUS

Adapter Administration

WebFOCUS Reporting Server Release 8206
DataMigrator Server Release 7710

January 30, 2020

Contents

Preface

This manual describes the configuration and management of Adapters. It is intended for database administrators and application developers.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Introduction to Adapters	Describes how to use adapters.
2	Using the Adapter for 1010data	Describes how to configure the Adapter for 1010data.
3	Using the Adapter for Adabas	Describes how to configure the Adapter for Adabas.
4	Using the Adapter for Adabas Stored Procedures	Describes how to configure the Adapter for Adabas Stored Procedures.
5	Using the Adapter for Alchemy Sentiment Analysis	Describes how to configure the Alchemy Sentiment Analysis Adapter.
6	Using the Adapter for Amazon Athena	Describes how to configure the adapter for Amazon Athena.
7	Using the Adapter for Amazon Redshift	Describes how to configure the Adapter for Amazon Redshift.
8	Using the Adapter for Apache Drill	Describes how to configure the adapter for Apache Drill.
9	Using the Adapter for Apache Hive to Access Data Managed by Hadoop	Describes how to configure the adapter for Hadoop with Apache Hive.
10	Using the Adapter for Apache Phoenix to Access HBase	Describes how to configure the adapter for Apache Phoenix to access HBase.
11	Using the Adapter for Apache Spark	Describes how to configure the Adapter for Apache Spark.
12	Using the Adapter for Axiom EPM	Describes how to configure the Adapter for Axiom EPM.

	Chapter/Appendix	Contents
13	Using the Adapter for C9 INC	Describes how to configure the Adapter for C9 INC.
14	Using the Adapter for Caché	Describes how to configure the Adapter for Caché.
15	Using the Adapter for CICS Transactions	Describes how to configure the Adapter for CICS Transactions.
16	Using the Adapter for Cloudera Impala to Access Data Managed by Hadoop	Describes how to configure the Adapter for Cloudera Impala.
17	Using the Adapters for C-ISAM and ISAM	Describes how to configure the Adapter for C-ISAM and ISAM.
18	Using the Adapter for DATACOM	Describes how to configure the Adapter for DATACOM.
19	Using the Adapter for Db2	Describes how to configure the Adapter for Db2.
20	Using the Adapter for DB Heritage Files	Describes how to configure the Adapter for DB Heritage Files.
21	Using the Adapter for Esri ArcGIS	Describes how to configure the adapter for Esri ArcGIS.
22	Using the Adapter for Essbase	Describes how to configure the Adapter for Essbase.
23	Using the Adapter for EXASol	Describes how to configure the adapter for EXASol.
24	Using the Adapter for Excel	Describes how to configure the Adapter for Excel.
25	Using the Adapter for Excel (via Direct Retrieval)	Describes how to configure the Adapter for Excel (via Direct Retrieval).
26	Using the Adapter for Facebook	Describes how to configure the Facebook Adapter.
27	Using the Adapters for Fixed-Format and Delimited Files	Describes how to configure the Adapter for Fixed-Format and Delimited Files.
28	Using the Adapter for Git	Describes how to configure the adapter for Git.

	Chapter/Appendix	Contents
29	Using the Adapter for Google Analytics	Describes how to configure the Google Analytics Adapter.
30	Using the Adapter for Google BigQuery	Describes how to configure the Google BigQuery Adapter.
31	Using the Adapter for Greenplum	Describes how to configure the Adapter for Greenplum.
32	Using the Adapter for HP Vertica	Describes how to configure the Adapter for HP Vertica.
33	Using the Adapter for Hyperledger Fabric	Describes how to configure and create synonyms for the adapter for Hyperledger Fabric.
34	Using the Adapter for Hyperstage	Describes how to configure the Adapter for Hyperstage.
35	Using the Adapter for i Access	Describes how to configure the Adapter for ODBC.
36	Using the Adapter for CA-IDMS/DB	Describes how to configure the Adapter for CA-IDMS/DB.
37	Using the Adapter for CA-IDMS/SQL	Describes how to configure the Adapter for CA-IDMS/SQL.
38	Using the Adapter for IMS	Describes how to configure and use the Adapter for IMS
39	Using the Adapter for IMS Transactions	Describes how to configure the Adapter for IMS Transactions.
40	Using the Adapter for Informix	Describes how to configure the Adapter for Informix.
41	Using the Adapter for Ingres	Describes how to configure the Adapter for Ingres.
42	Using the Adapter for Interplex	Describes how to configure the Adapter for Interplex.
43	Using the Adapter for iWay Adapter Framework (IWAF)	Describes how to configure the Adapter for iWay Adapter Framework (IWAF).

	Chapter/Appendix	Contents
44	Using the Adapter for JBoss Application Server	Describes how to configure the Adapter for JBoss Application Server.
45	Using the Adapter for JDBC	Describes how to configure the Adapter for JDBC.
46	Using the Adapter for JD Edwards EnterpriseOne	Describes how to configure the Adapter for JD Edwards EnterpriseOne.
47	Using the Adapter for JD Edwards World	Describes how to configure the Adapter for JD Edwards World.
48	Using the Adapter for Jethro	Describes how to configure the Adapter for Jethro.
49	Using the Adapter for JSON	Describes how to configure the Adapter for JSON.
50	Using the Adapter for Kafka	Describes how to configure the Adapter for Kafka.
51	Using the Adapter for Lawson	Describes how to configure the Adapter for Lawson.
52	Using the Adapter for LinkedIn	Describes how to configure the LinkedIn Adapter.
53	Using the Adapter for Lotus Notes	Describes how to configure the Adapter for Lotus Notes.
54	Using the Adapter for LDAP	Describes how to configure the Adapter for LDAP.
55	Using the Adapter for MariaDB	Describes how to configure the Adapter for MariaDB.
56	Using the Adapter for Microsoft Access	Describes how to configure the Adapter for Microsoft Access.
57	Using the Adapter for Microsoft Azure SQL Data Warehouse	Describes how to configure the Adapter for Microsoft Azure SQL Data Warehouse.
58	Using the Adapter for Microsoft Dynamics CRM	Describes how to configure the Adapter for Microsoft Dynamics CRM
59	Using the Adapter for Microsoft SQL Server	Describes how to configure the Adapter for Microsoft SQL Server.

	Chapter/Appendix	Contents
60	Using the Adapter for Microsoft SQL Server Analysis Services (SSAS)	Describes how to configure the Adapter for SQL Server Analysis Services (SSAS).
61	Using the Adapter for Microsoft SQL Server Analysis Services Tabular Data Model	Describes how to configure the Adapter for Microsoft SQL Server Analysis Services Tabular Data Model.
62	Using the Adapter for Microsoft SQL Server ODBC	Describes how to configure the Adapter for Microsoft SQL Server ODBC.
63	Using the Adapter for Millennium	Describes how to configure the Adapter for Millennium.
64	Using the Adapter for Model 204	Describes how to configure the Adapter for Model 204.
65	Using the Adapter for MongoDB	Describes how to configure the adapter for MongoDB.
66	Using the Adapter for MySQL	Describes how to configure the Adapter for MySQL.
67	Using the Adapter for NATURAL	Describes how to configure the Adapter for NATURAL.
68	Using the Adapter for NATURAL CICS Transactions	Describes how to configure the Adapter for NATURAL CICS Transactions.
69	Using the Adapter for Netezza	Describes how to configure the Adapter for Netezza.
70	Using the Adapter for Nucleus	Describes how to configure the Adapter for Nucleus.
71	Using the Adapter for OData	Describes how to configure the Adapter for OData.
72	Using the Adapter for ODBC	Describes how to configure the Adapter for ODBC.
73	Using the Adapter for Oracle	Describes how to configure the Adapter for Oracle.
74	Using the Adapter for Oracle E-Business Suite	Describes how to configure the Adapter for Oracle E-Business Suite.

	Chapter/Appendix	Contents
75	Using the Adapter for Oracle TimesTen	Describes how to configure the Adapter for Oracle TimesTen.
76	Using the Adapter for parAccel	Describes how to configure the Adapter for parAccel.
77	Using the Adapter for PeopleSoft	Describes how to configure the Adapter for PeopleSoft.
78	Using the Adapter for PostgreSQL	Describes how to configure the Adapter for PostgreSQL.
79	Using the Adapter for Presto	Describes how to configure the Adapter for Presto
80	Using the Adapter for Progress	Describes how to configure the Adapter for Progress.
81	Using the Adapter for PSQL	Describes how to configure the Adapter for PSQL.
82	Using the Adapter for Python	Describes how to configure the adapter and run Python scripts.
83	Using the Adapter for Query/400	Describes how to configure the Adapter for Query/400.
84	Using the Adapter for Rdb	Describes how to configure the Adapter for Rdb.
85	Using the Adapter for Remote Servers	Describes how to configure the Adapter for Remote Servers.
86	Using the Adapter for REST	Describes how to configure the Adapter for REST.
87	Using the Adapter for RMS	Describes how to configure the Adapter for RMS.
88	Using the Adapter for Rserve	Describes how to configure the adapter, create a synonym for an R script, and run the script using the RSERVE function.
89	Using the Adapter for Salesforce.com	Describes how to configure the Adapter for Salesforce.com.

	Chapter/Appendix	Contents
90	Using the Adapter for SAP Business Intelligence Warehouse (BW)	Describes how to configure the Adapter for SAP Business Intelligence Warehouse (BW).
91	Using the Adapter for SAP ERP	Describes how to configure the Adapter for SAP.
92	Using the Adapter for SAP Hana	Describes how to configure the Adapter for SAP Hana.
93	Using the Adapter for Siebel	Describes how to configure the Adapter for Siebel.
94	Using the Adapter for Slack	Describes how to configure the Adapter for Slack.
95	Using the Adapter for Snowflake Cloud Data Warehouse	Describes how to configure the Adapter for Snowflake Cloud Data Warehouse
96	Using the Adapter for SQLBase	Describes how to configure the Adapter for SQLBase.
97	Using the Adapter for Stratio Crossdata	Describes how to configure the Adapter for Stratio Crossdata.
98	Using the Adapter for Supra	Describes how to configure the Adapter for Supra
99	Using the Adapter for Sybase	Describes how to configure the Adapter for Sybase.
100	Using the Adapter for Teradata	Describes how to configure the Adapter for Teradata.
101	Using the Adapter for Transoft	Describes how to configure the Adapter for Transoft.
102	Using the Adapter for Twitter	Describes how to configure the Twitter Adapter.
103	Using the Adapter for UniData	Describes how to configure the Adapter for UniData.
104	Using the Adapter for UniVerse	Describes how to configure the Adapter for UniVerse.
105	Using the Adapter for VSAM	Describes how to configure the Adapter for VSAM.

	Chapter/Appendix	Contents
106	Using the Adapter for WAND Sentiment Analysis	Describes how to configure the WAND Sentiment Analysis Adapter.
107	Using the Adapter for Web Services	Describes how to configure the Adapter for Web Services.
108	Using the Adapter for WebFOCUS Client REST	
109	Using the Adapter for Words Analysis	Describes how to configure the Words Analysis Adapter.
110	Using the Adapter for XML	Describes how to configure the Adapter for XML.
A	XA Support	Describes XA Transaction Management and implementation specifics.
B	Aggregate Awareness Support	Describes Aggregate Awareness, which substantially improves the efficiency of queries.
C	Cluster Join	Describes the Master File SEGSUF attribute, which enables you to create heterogeneous cluster joins.
D	Translating COBOL File Descriptions	Describes how synonyms are created from COBOL File Descriptions.
E	Data Set Compression Exit: ZCOMP	Describes how to use the ZCOMP Exit for data set compression.
F	Dynamic Private User Exit	Describes how use the Dynamic Private User Exit API.
G	Validation for Special Characters and Reserved Words	Describes how the Web Console validates special characters and reserved words.

Conventions

The following conventions apply throughout this manual:

Convention	Description

`THIS TYPEFACE`
or
`this typeface`
	Denotes syntax that you must enter exactly as shown.

`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices. Type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our Technical Documentation Library at *http://documentation.informationbuilders.com*. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques, *http://forums.informationbuilders.com/eve/forums*.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, *http://www.informationbuilders.com*. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- ❏ Your six-digit site code (*xxxx.xx*).
- ❏ Your Server Software configuration:
  - ❏ The version and release. You can find your server version and release using the *Version* option in the Web Console.
  - ❏ The communications protocol (for example, TCP/IP), including vendor and release.
- ❏ The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- ❏ The database server release level.

- The database name and release level.
- The Master File and Access File.
- The exact nature of the problem:
  - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
  - The error message and return code, if applicable.
  - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. You can contact us through our website *http://documentation.informationbuilders.com/connections.asp*.

Thank you, in advance, for your comments.

Information Builders Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website (*http://education.informationbuilders.com*) or call (800) 969-INFO to speak to an Education Representative.

Chapter 1

Introduction to Adapters

In client/server architecture, adapters enable clients to manage data from virtually any data source, and on any operating system, through the use of SQL statements.

The client generates requests for data residing on the server. The server acts as a source of data, and can accept requests from multiple clients for data access and manipulation. Client/server architecture divides a traditional single system into a front-end and a back-end. The workload is distributed between the client and the server. Communications software establishes the link between client and server, and interfaces to the desired communications protocol.

In this chapter:

Processing Requests

The server can process SQL requests. This enables it to support a wide variety of client applications that use SQL and for those clients, SQL is the standard access language for requests to all relational and non-relational data.

In addition, the server has its own 4-GL request language called WebFOCUS that provides a much more robust set of features than SQL and extends the types of requests that can be processed, especially requests against non-relational data sources whose architecture is not based on the relational model.

The server in client/server architecture is used for data access and manipulation. The server receives a request for data, processes it, and returns an answer set or message to the client. Adapters are among the various subsystems that comprise a server.

Functions of an Adapter

The server uses adapters to access data sources. The server can receive two types of requests from the client: SQL or WebFOCUS.

When the server receives requests from the client, it passes them to the adapter in a standard format. The adapter takes the request, transforms it into the native data manipulation language (DML), and then issues calls to the data source using its API. In this way, the adapter insulates the server from details of the data source. An application can issue SQL statements, issue WebFOCUS commands, or call stored procedures.

Adapters are available for many data sources. Every adapter is specifically designed for the data source that it accesses, and, as a result, is able to translate between SQL or WebFOCUS and the DML of the data source. Adapters provide solutions to product variations, including product differences in syntax, functionality, schema, data types, catalogs, data representations, message processing, and answer set retrieval.

Note: If the adapter has read/write capabilities, it inserts the data from an application to the data source.

How an Adapter Works

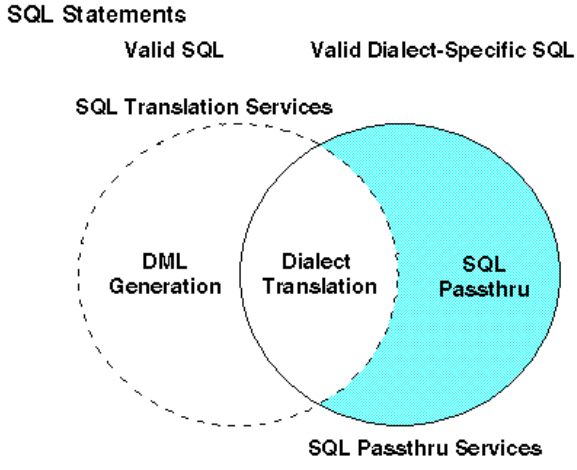
The adapter manages the communication between the data interface and the data source, passing data management requests to the data source and returning either answer sets or messages to the requestor.

To perform these functions, the adapter:

1. Translates the request to the applicable DML.
2. Attaches to the targeted data source, using standard attachment calls. The adapter then passes the request to the data source.
3. The data source processes the request.
4. The results or error conditions are returned to the client application for further processing.

Processing SQL Requests

A server can be configured to behave in different ways upon receipt of SQL requests from a client application. A server handles SQL requests for data in the following ways, as illustrated in the diagram:



- **Direct Passthru.** When Direct Passthru is enabled, the server passes SQL requests directly to the specified RDBMS for processing. The name of the targeted RDBMS (the database engine) is supplied in the server profile or, in some cases, by the client application. A Full-Function or Hub Server can operate temporarily in Direct Passthru mode when invoked by a client application. The user is responsible for activating and deactivating Direct Passthru as needed.
- **SQL Processing.** When the database engine is not set in the server profile or supplied by a client application, Direct Passthru is not enabled. Instead, a Hub or Full-Function Server invokes its default behavior: it accepts the incoming SQL request and verifies that it is valid. Then the server determines if it can process the incoming SQL request:
  - If the request meets certain requirements, the server passes it directly to the RDBMS for processing. This is called Automatic Passthru.
  - If the syntax of the request does not conform to the syntax of the RDBMS, the server translates the request into internal DML and passes it to the adapter. The adapter generates adapter-specific DML and passes the request to the RDBMS for processing. This is called SQL translation.

Processing WebFOCUS Requests

When the server receives a WebFOCUS request, it passes it to the adapter. The adapter analyzes the request and generates DBMS-specific DML for those parts of the request that have DML equivalents. As WebFOCUS is a more robust language than DML, some parts of the request may not have such equivalents. The adapter then passes the generated DML to the DBMS for processing. When the answer set is returned by the DBMS, WebFOCUS processes those parts of the request that the adapter could not pass to the DBMS.

Throughout this manual there are descriptions of some useful features that are not supported by SQL syntax but are supported by WebFOCUS syntax.

For complete information about the WebFOCUS language, see the following manuals:

- *Creating Reports With WebFOCUS Language*
- *Describing Data With WebFOCUS Language*
- *Developing Reporting Applications*

Relational and Non-Relational Adapters

Adapters can retrieve answer sets from both relational and non-relational data sources. Since the architectures of relational and non-relational data structures vary, the relational and non-relational adapters adjust for these differences. For example, relational adapters are designed to handle data sources that contain data in rows and columns in tables, while non-relational adapters are designed to accommodate the architecture of each distinct data source, for instance, a hierarchical or network data source, or a sequential or indexed file system.

The following table lists key features of relational and non-relational adapters:

Feature	Relational Adapters	Non-Relational Adapters
DEFINE (virtual field) in Master File	Yes	Yes
Access Control	Yes	Yes
Transaction Management Commands	Yes	No

Relational and non-relational adapters:

- Allow the data source to perform the work required to join, sort, and aggregate data. Therefore, the volume of data source-to-server communication is reduced, resulting in better response times for users. The adapter tries to optimize the queries as best it can.

- Communicate with the data source through DML statements. You can view these DML statements with traces provided by the trace facilities. These traces are helpful for debugging your procedure or for adapter performance analysis. Note that traces:
  - Are common for all relational adapters.
  - Vary for non-relational adapters to accommodate the differences in data structures and DML calls.
- Support both DEFINE (virtual) fields and DBA.
- Relational adapters include a variety of COMMIT, connection, and thread control commands that enable Database Administrators to control the opening and closing of connections and choose when to commit or rollback transactions. This level of transaction control is not supported by non-relational adapters. In Full-Function Server mode, COMMIT/ROLLBACK will be propagated to all relational data sources local to the server. If the hub is active, a COMMIT will be issued against remote data servers as well. All PREPARE handles are cleared (this is a Broadcast COMMIT).

To address these similarities and differences, this manual contains:

- *General* components with information that is common to all adapters, as well as information that is common either to all relational or to all non-relational adapters.
- *Customized* components with information that applies to specific adapters.

Supported Adapters

For the most current and specific information about releases and platforms,

1. Go to *http://techsupport.informationbuilders.com*.

   The Information Builders Technical Support home page opens.

2. In the Quick Links section on the right side of the page, click *Supported Systems/Adapters*.

   The Supported Systems and Adapters page opens.

3. Click the link for the server release you want.

   The Supported Systems and Adapters page for that release opens.

4. Click the link for your platform.

   The support chart for that platform opens.

The following table summarizes that information.

Adapter	Windows	UNIX	z/OS	OpenVMS	IBM i (formerly i5/OS)
1010data	x	x			
Adabas	x	x	x		
Adabas Stored Procedures			x		
Address Doctor	x	x	x	x	x
Alchemy	x	x	x	x	x
Business Object Universe	x	x	x	x	x
Caché	x	x			
CA-IDMS/DB			x		
CA-IDMS/SQL			x		
CICS Transactions	x	x	x	x	x
C-ISAM	x	x			
DATACOM/DB			x		
Db2	x	x	x		x
DB Heritage Files					x
Delimited Flat Files	x	x	x	x	x
Essbase	x	x			
Excel	x				
Excel (via direct retrieval)	x	x	x	x	x
Facebook	x	x	x	x	x
Flat Files	x	x	x	x	x
FOCUS	x				
Greenplum	x	x	x	x	x

Adapter	Windows	UNIX	z/OS	OpenVMS	IBM i (formerly i5/OS)
Hive/Hadoop Release Candidate	x	x	x	x	x
Hyperstage	x	x	x	x	x
IDMS > see CA-IDMS					
IMS/DB			x		
IMS Transactions	x	x	x	x	x
Information Manager			x		
Informix	x	x			
Ingres		x			
Interplex(DMS/RDMS 2200/1100)	x	x			
iWay Adapter Framework (IWAF)	x	x	x		
JBoss	x	x	x	x	x
JDBC	x	x	x	x	x
JD Edwards EnterpriseOne	x	x	x	x	x
JD Edwards World					x
JSON	x	x	x	x	x
Lawson	x	x	x	x	x
LDAP	x	x	x		x
Lotus Notes	x	x	x		x
Microsoft Access	x				
Microsoft SQL Server	x	x	x	x	x*

Adapter	Windows	UNIX	z/OS	OpenVMS	IBM i (formerly i5/OS)
Millennium			x		
Model 204			x		
MySQL	x	x	x	x	x
NATURAL Batch	x	x	x	x	x
NATURAL CICS Transactions		x	x		x
Netezza	x				
Nucleus	x	x			
ODBC	x				
Oracle	x	x		x	
Oracle E-Business Suite (Oracle Applications)	x	x	x	x	
parAccel	x				
PeopleSoft	x	x	x		
PostgreSQL	x	x	x	x	x*
Progress	x	x			
PSQL	x	x	x	x	x*
Query/400					x
Rdb				x	
REST	x	x	x	x	x
Remote Servers	x	x	x	x	x
RMS				x	
SAP BW	x	x	x		x

Adapter	Windows	UNIX	z/OS	OpenVMS	IBM i (formerly i5/OS)
SAP	x	x	x (non-Unicode)		x
Siebel	x	x			x*
SQLBase	x	x	x	x	x*
SQL Server Analysis Services (SSAS)	x				
Sybase	x	x			
Teradata	x	x	x		
Transoft	x	x	x	x	x*
Twitter	x	x	x	x	x
Unidata	x	x	x	x	x*
UniVerse	x	x			
VSAM			x		
VSAM CICS			x		
WAND	x	x	x	x	x
Web Services	x	x	x	x	x
Words Analysis	x	x	x	x	x
XML	x	x	x	x	x

* The underlying databases for these adapters do not necessarily run directly on the IBM i platform (formerly known as i5/OS). However, they may be configured and accessed.

Data Management

As you manage your data, you may be required to modify your server and communications configuration files. The first step is understanding how and where data is described and the roles of the server and adapters in managing the processing flow.

Describing Data Sources

In order to access a table or view, you must first describe it using two files: a Master File and an associated Access File.

Master Files and Access Files can represent an entire table or part of a table. Also, several pairs of Master and Access Files can define different subsets of columns for the same table, or one pair of Master and Access Files can describe several tables.

Note: In these topics, the term table refers to both base tables and views in data sources. The Master File describes the columns of the data source table using keywords in comma-delimited format. The Access File includes additional parameters that complete the definition of the data source table. Some adapters require both files to fulfill queries, and to build the DML to access the non-SQL data sources.

Processing Requests

When requests are processed, control is passed from the server to an adapter and back. During the process, selected information is read from the Master and Access Files as described below.

The server processes a request as follows:

1. The request is parsed to identify the table.
2. The Master File for the table is read.
3. The SUFFIX value in the Master File is checked (SUFFIX indicates the type of data source).
4. Control is passed to the appropriate adapter.

The adapter then:

1. Locates the corresponding Access File.
2. Uses the information contained in the Master and Access Files to generate the DML statements (if necessary) required to accomplish the request.
3. Passes the DML statements to the data source.
4. Retrieves the answer set generated by the data source.
5. Returns control to the server.

Depending on the requirements of the request, additional processing may be performed on the returned data.

Master File

A Master File describes a logical data source. A logical data source can be made up of one or more physical data sources of the same type. Each segment is a physical data source.

Master Files contain three types of declarations:

Declaration Type	Description
File	Names the file and describes the type of data source.
Segment	Identifies a table, file, view, or segment.
Field	Describes the columns of the table, view, or fields in the file.

The following guidelines apply:

- A declaration consists of attribute-value pairs separated by commas.
- Each declaration must begin on a separate line. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.
- Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

Syntax: How to Specify a File Declaration in a Master File

A Master File begins with a file declaration, which has at least two attributes:

`FILENAME (FILE)`

Identifies the Master File.

`SUFFIX`

Identifies the adapter needed to interpret the request.

The syntax for a file declaration is

`FILE[NAME]=`*`file`*`, SUFFIX=`*`suffix`* `[,$]`

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters.

suffix

Identifies the adapter needed to interpret the request. For example, SQLORA is the value for the Adapter for Oracle.

Syntax: How to Specify a Segment Declaration in a Master File

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes:

SEGNAME

Identifies one table.

SEGTYPE

Identifies the physical storage of rows and the uniqueness of column values.

The syntax for a segment declaration is

```
SEGNAME=segname, SEGTYPE=S0 [,$]
```

where:

segname

Is the segment name which serves as a link to the actual table name. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name. It can consist of a maximum of 8 alphanumeric characters.

The SEGNAME value in the Master File must be the same as the SEGNAME value specified in the Access File, where the TABLENAME portion of the segment declaration contains the fully-qualified name of the table.

S0

Indicates that the RDBMS is responsible for both physical storage of rows and the uniqueness of column values (if a unique index or constraint exists). It always has a value of S0 (S zero).

Syntax: **How to Specify a Field Declaration in a Master File**

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

`FIELDNAME`

Identifies the name of a field.

`ALIAS`

Identifies the full column name.

`USAGE`

Identifies how to display a field on reports.

`ACTUAL`

Identifies the data type and length in bytes for a field.

`MISSING`

Identifies whether a field supports null data.

You can obtain values for these attributes by using the system catalog table ALL_TAB_COLUMNS for the existing table or view you wish to describe.

The syntax for a field declaration is

```
FIELD[NAME]=fieldname, [ALIAS=]sqlcolumn, [USAGE=]display_format,
     [ACTUAL=]storage_format [,MISSING={ON|OFF}], $
```

where:

fieldname

Is the name of the field. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters including letters, digits, and underscores. The name must begin with a letter. Special characters and embedded blanks are not recommended. The order of field declarations in the Master File is significant with regard to the specification of key columns. For more information, see *Primary Key* on page 88.

Tip: Since the name appears as the default column title for reports, for client applications, or EDADESCRIBE, select a name that is representative of the data.

It is not necessary to describe all the columns of the table in your Master File.

sqlcolumn

Is the full column name (the adapter uses it to generate SQL statements). This value must comply with the naming conventions for identifiers, where a name should start with a letter and may be followed by any combination of letters, digits, or underscores. Embedded spaces are not allowed.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column *includes* the decimal point and negative sign. In SQL, the total length of the field or column *excludes* the decimal point and negative sign. For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

storage_format

Is the storage format of the data type and length in bytes.

ON

Displays the character specified by the NODATA parameter for missing data. For related information, see *MISSING Attribute* on page 86.

OFF

Displays blanks or zeroes for fields having no value. This is the default. For more information, see *MISSING Attribute* on page 86.

MISSING Attribute

In a table, a null value represents a missing or unknown value; it is not the same as a blank or a zero. For example, a column specification that allows null values is used where a column need not have a value in every row (such as a raise amount in a table containing payroll data).

- The default NODATA character is a period.
- A column in a table that allows null data does not need to include the NULL clause in its table definition, since that is the default.

- In the Master File for that table, the column that allows null data must be described with the MISSING attribute value ON. The default for this attribute is OFF, which corresponds to the NOT NULL attribute in the table definition.

If the column allows null data but the corresponding field in the Master File is described with the MISSING attribute value OFF, null data values appear as zeroes or blanks.

Access File

Each Master File may have a corresponding Access File. The name of the Access File must be identical to that of the Master File, but the extension will be .acx instead of .mas.

The Access File serves as a link between the server and the data source by providing the means to associate a segment in the Master File with the table it describes. The Access File minimally identifies the table and primary key (if there is one). It may also indicate the logical sort order of data and identify storage areas for the table.

Syntax: How to Specify a Segment Declaration in an Access File

The segment declaration in the Access File establishes the link between one segment of the Master File and the actual table or view. Attributes that constitute the segment declaration are:

`SEGNAME`

Identifies one table.

`TABLENAME`

Identifies the table or view. It may contain the owner ID as well as the table name.

`KEYS`

Identifies how many columns constitute the primary key.

`KEYORDER`

Identifies the logical sort sequence of data by the primary key.

The syntax for a segment declaration in an Access File is

```
SEGNAME=segname, TABLENAME=owner.tablename databaselink
   [,KEYS={n|0}] [,KEYORDER={LOW|HIGH}] ,$
```

where:

segname

Is the same value as the SEGNAME value in the Master File.

owner

Is the user ID by default.

tablename

Is the name of the table or view.

databaselink

Is the DATABASE LINK name to be used in the currently connected database server.

n

Is the number of columns that constitute the primary key. It can be a value from 0 to 16. The default value is 0. For more information, see *Primary Key* on page 88.

LOW

Indicates an ascending primary key logical sort order. This value is the default.

HIGH

Indicates a descending primary key logical sort order.

Primary Key

A primary key consists of the column or combination of columns whose values uniquely identify each row of the table. In the employee table, for example, every employee is assigned a unique employee identification number. Each employee is represented by one and only one row of the table, and is uniquely identified by that identification number.

The primary key definition must be defined partly in the Master File and partly in the Access File:

- The order of field declarations in the Master File is significant to the specification of key columns. To define the primary key in a Master File, describe its component fields immediately after the segment declaration. You can specify the remaining fields in any order. In the Access File, the KEYS attribute completes the process of defining the primary key.
- To identify the primary key, the adapter uses the number of columns (*n*) indicated by the KEYS attribute in the Access File and the first *n* fields described in the Master File.

Typically, the primary key is supported by the creation of a unique index in the SQL language to prevent the insertion of duplicate key values. The adapter itself does not require any index on the column(s) comprising the primary key (although a unique index is certainly desirable for both data integrity and performance reasons).

Creating Virtual Fields

You use the DEFINE command to accomplish these tasks.

Syntax: How to Create Virtual Fields With the DEFINE Command

```
DEFINE fieldname/format [WITH fieldname]=expression ;$
```

where:

fieldname

Is a field name for the virtual field. It can consist of 1 to 48 characters. You must not qualify the field name.

format

Provides the display format for the field and follows the rules for USAGE formats. This operand is optional. If not specified, the default value is D12.2.

WITH *fieldname*

Must be coded when the expression is a constant. Any real field can be chosen from the same segment the DEFINE is associated with.

expression

Can be either a mathematical or a logical statement. It can consist of constants, database fields, and virtual fields. The expression must end with a semicolon followed by a dollar sign (;$).

Place your DEFINE statements after all of the field descriptions in the segment. If you are using the DESCRIPTION or TITLE attributes with virtual fields, you must place these attributes on a separate line.

Example: Defining a Virtual Field in a Master File

In the example that follows, the virtual field PROFIT is defined at the end of the segment named BODY.

```
SEGMENT=BODY, SEGTYPE=S0 , PARENT=CARREC,$
 FIELDNAME=BODYTYPE          ,ALIAS=BODYTYPE      ,A12,A12,$
FIELDNAME=DEALER_COST        ,ALIAS=DEALER_COST   ,D8, D8 ,$
FIELDNAME=RETAIL_COST        ,ALIAS=RETAIL_COST   ,D8, D8 ,$
DEFINE PROFIT/D8 = RETAIL_COST - DEALER_COST
 ;DESC=NET_COST, TITLE='NET,COST' ,$
```

As a result of this DEFINE statement, you can use PROFIT as a field name in reports. PROFIT is treated as a field with a value equal to the value of RETAIL_COST minus DEALER_COST.

Note:

- Since the complete data source needs to be read to calculate virtual fields, screening conditions on virtual fields may incur additional overhead.
- Virtual fields in the Master File for relational and remote data sources will, if referenced in a query, disable Automatic Passthru.

Cross-Century Dates

Many existing business applications use two digits to designate a year, instead of four digits. When they receive a value for a year, such as 00, they typically interpret it as 1900, assuming that the first two digits are 19, for the twentieth century. There is considerable risk that date-sensitive calculations in existing applications will be wrong unless an apparatus is provided for determining the century in question. This will impact almost every type of application, including those that process mortgages, insurance policies, anniversaries, bonds, inventory replenishment, contracts, leases, pensions, receivables, and customer records.

The cross-century dates feature enables you to solve this problem at the file and field level of your applications. You can retain your global settings while changing the file-level settings for greater flexibility.

You can enable this feature:

- Using SET commands.
- At the file level in a Master File.
- At the field level in a Master File.

Cross-Century Dates SET Commands

The server delivers SET commands that provide a means of interpreting the century if the first two digits of the year are not provided:

```
SET DEFCENT
SET YRTHRESH
```

If the first two digits are provided, they are simply accepted and validated.

Syntax: How to Implement a Cross-Century Date

The DEFCENT syntax is

```
SET DEFCENT=nn
```

where:

nn

Is 19 unless otherwise specified.

The YRTHRESH syntax is

```
SET YRTHRESH=nn
```

where:

nn

Is zero unless otherwise specified.

The combination of DEFCENT and YRTHRESH establishes a base year for a 100-year window. Any 2-digit year is assumed to fall within that window, and the first two digits are set accordingly. Years outside the declared window must be handled by user coding.

The default values for the two commands are SET DEFCENT=19, SET YRTHRESH=00. When you provide a year threshold, years greater than or equal to that value assume the value assigned by DEFCENT. Years lower than that threshold become DEFCENT plus 1.

To see how DEFCENT and YRTHRESH are applied to interpret 2-digit years, consider the following:

```
SET DEFCENT=19, SET YRTHRESH=80
```

This set of commands describes a range from 1980 to 2079. If a 2-digit year field contains the value 99, then the server interprets the year as 1999. If the year field is 79, then the year is interpreted as 2079. If the year field is 00, then the year is interpreted as 2000.

Master File Syntax

Instead of using SET commands, you can include settings at the file level in a Master File, or at the field level in a Master File.

Syntax: How to Add Cross-Century Date Settings at the File Level

The FDEFCENT syntax is

```
{FDEFCENT|FDFC}=nn
```

where:

nn

Is 19, unless otherwise specified.

The FYRTHRESH syntax is

```
{FYRTHRESH|FYRT}=nn
```

where:

nn

Is zero, unless otherwise specified.

Syntax: How to Add Cross-Century Date Settings at the Field Level

At the field level, DEFCENT and YRTHRESH can be added. The DEFCENT syntax is

```
{DEFCENT|DFC}=nn
```

where:

nn

Is 19, unless otherwise specified.

The YRTHRESH syntax is

```
{YRTHRESH|YRT}=nn
```

where:

nn

Is zero, unless otherwise specified.

Syntax: How to Add Cross-Century Dates Using a DEFINE Command

```
DEFINE FILE EMPLOYEE
 fld/fmt [{DEFCENT|DFC} nn {YRTHRESH|YRT} nn] [MISSING...]=expression;
END
```

The DFC and YRT syntax must follow the field format information.

Example: Implementing Cross-Century Dates

The following example illustrates how century interpretation is implemented at both the file level and field level in a Master File.

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDEFCENT=20, FYRTHRESH=66,$
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,     ALIAS=EID,  FORMAT=A9,       $
  FIELDNAME=LAST_NAME,  ALIAS=LN,   FORMAT=A15,      $
  FIELDNAME=FIRST_NAME, ALIAS=FN,   FORMAT=A10,      $
  FIELDNAME=HIRE_DATE,  ALIAS=HDT,  FORMAT=I6YMD, DEFCENT=19,
YRTHRESH=75,$
```

The next example illustrates the conversion of a 2-digit year field with the DEFINE command:

```
DEFINE FILE EMPLOYEE
ESHIRE_DATE/YYMD = HIRE_DATE; (The format of HIRE_DATE is I6YM.)
ESHIRE DFC 19 YRT 80 = HIRE_DATE;
END
```

Reference: **Synonym Management Options**

You can right-click a synonym in the Adapter navigation pane of either the Web Console or the Data Management Console to access the following options.

Option	Description
Open	Opens the Master File for viewing and editing using a graphical interface. If an Access file is used it will be also available.
Edit as Text	Enables you to view and manually edit the Master File synonym. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the *Open* option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to view and manually edit the Access File synonym. **Note:** This option is available only when an Access File is created as part of the synonym.
Sample Data	Retrieves up to 20 rows from the associated data source.
Data Profiling	Data Profiling provides the data characteristics for synonym columns. Alphanumeric columns provide the count of distinct values, total count, maximum, minimum, average length, and number of nulls. Numeric columns provide the count of distinct values, total count, maximum, minimum, average value, and number of nulls.
Refresh Synonym (if applicable)	Regenerates the synonym. Use this option if the underlying object has been altered.

Option	Description
Data Management	Followed by these options, if applicable: **Recreate DBMS Table.** Recreates the data source table. You are asked to confirm this selection before the table is regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.) **Delete All Data.** Deletes all existing data. You are asked to confirm this selection before the data is deleted. **Drop Table.** Drops the table so that it is removed from the DBMS. **Insert Sample Data.** Inserts specified number of sample records, populating all fields with counter values. **Show/Modify Data.** Opens a window that shows the data in the data source with buttons you can click to insert values, filter values, reload the data source, and customize the view. **Reorganize.** Recreates the data source table preserving original data. **Note:** This option is not available in the Web Console.
Impact Analysis	Generates a report showing where this synonym is stored and used, with links to the synonym instances. Impact Analysis reports enable you to evaluate changes before they are made by showing which components will be affected. See the *Server Administration* manual for details about Impact Analysis reports.
Dependencies Analysis	Generates a report showing information about the synonym and other synonyms and objects that are referenced within it.
Copy	Copies the synonym to the clipboard.
Delete	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Cut	Deletes the synonym and places it on the clipboard.
Privileges	Shows the security subjects on the server and the privileges they have to this synonym.

Option	Description
Properties	Displays the properties of the synonym, including physical location, last modified date, description, and privileges.

Data Type Support Report

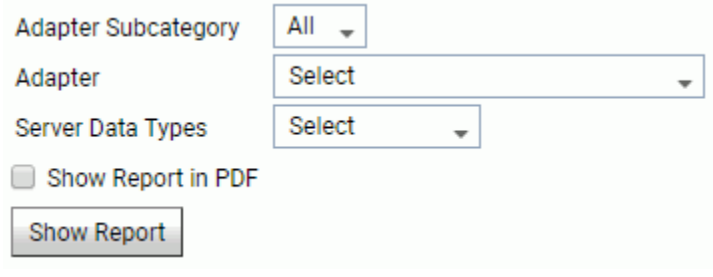
SQL Data Type mapping options are available in a report available from the Web Console.

Procedure: How to Access the Data Type Report

To access the Data Type Report:

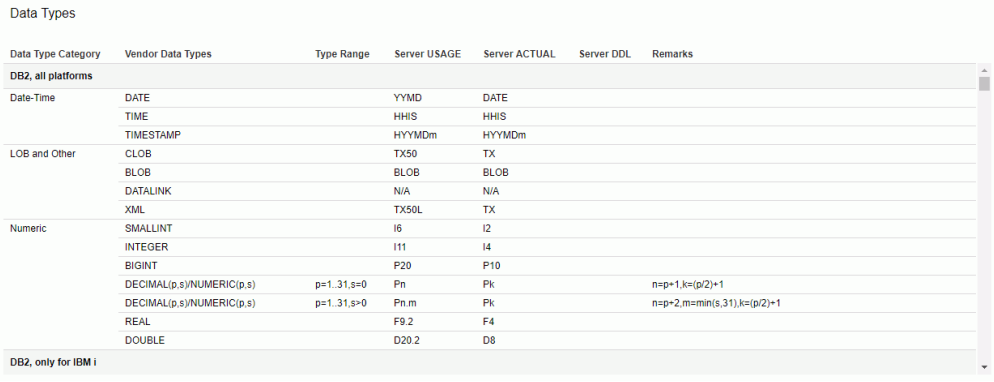
1. Select *Adapters* from the main menu.
2. From the Troubleshooting section of the Ribbon, click the *Data Types* button.

   The Filter Data Types Report page opens, as shown in the following image.



3. Select an *Adapter Subcategory* from the corresponding list box.
4. Select an *Adapter* from the corresponding list box.
5. Select a *Server Data Type* from the corresponding list box.
6. Click *Show Report*.

The Filter Data Types Report is displayed, as shown in the following image.



Data Types

Data Type Category	Vendor Data Types	Type Range	Server USAGE	Server ACTUAL	Server DDL	Remarks
DB2, all platforms						
Date-Time	DATE		YYMD	DATE		
	TIME		HHIS	HHIS		
	TIMESTAMP		HYYMDm	HYYMDm		
LOB and Other	CLOB		TX50	TX		
	BLOB		BLOB	BLOB		
	DATALINK		N/A	N/A		
	XML		TX50L	TX		
Numeric	SMALLINT		I6	I2		
	INTEGER		I11	I4		
	BIGINT		P20	P10		
	DECIMAL(p,s)/NUMERIC(p,s)	p=1..31,s=0	Pn	Pk		n=p+1,k=(p/2)+1
	DECIMAL(p,s)/NUMERIC(p,s)	p=1..31,s>0	Pn.m	Pk		n=p+2,m=min(s,31),k=(p/2)+1
	REAL		F9.2	F4		
	DOUBLE		D20.2	D8		
DB2, only for IBM i						

Note: You can also display the report as a PDF by selecting the *Show Report in PDF* check box.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

Syntax: How to Override the Default Precision and Scale

```
ENGINE ADAPTER_ID SET CONVERSION RESET
ENGINE ADAPTER_ID SET CONVERSION format RESET
ENGINE ADAPTER_ID SET CONVERSION format [PRECISION precision [scale]]
ENGINE ADAPTER_ID SET CONVERSION format [PRECISION MAX]
```

where:

ADAPTER_ID

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

The adapter values are:

Adapter	Adapter ID	Adapter	Adapter ID
Cache	dbsqlism	Db2	dbdb2
Excel	dbsqlexc	HP Vertica	dbsqlvrt
Hyperstage	dbsqlhyp	i Access	dbsqliia
CA-IDMS/DB	dbidmsr	Informix	dbsqlinf
Ingres	dbsqling	Interplex	dbsqlipx
JBoss Application Server	dbsqlmmx	JDBC	dbsqljdb
MariaDB	dbmariadb	Microsoft Access	dbsqlmac
Microsoft SQL Server	dbsqlmss	Microsoft SQL Server ODBC	dbmsodbc
MySQL	dbsqlmys	Netezza	dbsqlnez
Nucleus	dbsqlnuc	ODBC	dbsqlodb
Oracle	dbsqlora	Oracle TimesTen	dbsqlott
PostgreSQL	dbsqlpst	Progress	dbsqlpro
PSQL	dbsqlpsq	SAP Hana	dbsqlhan
Remote Servers	dbeda	Sybase	dbsqlsyb
SQLBase	dbsqlbas	Transoft	dbsqltrn
Teradata	dbsqldbc	UniVerse	dbsqluv
UniData	dbsqlund		
XML	dbxml		

RESET

Returns any previously specified precision and scale values to the adapter defaults. If you specify RESET immediately following the SET CONVERSION command, all data types return to the defaults. If you specify RESET following a particular data type, only columns of that data type are reset.

format

Is any valid format supported by the data source. Possible values are:

INTEGER which indicates that the command applies only to INTEGER columns.

DECIMAL which indicates that the command applies only to DECIMAL columns.

REAL which indicates that the command applies only to single-precision floating-point columns. Only applies to Db2, CA-IDMS/SQL, Microsoft SQL Server, and Sybase.

FLOAT which indicates that the command applies only to double-precision floating-point columns.

precision

Is the precision. Must be greater than 1 and less than or equal to the maximum allowable value for the data type (see the description of MAX).

scale

Is the scale. This is valid with DECIMAL, FLOAT and REAL data types. If you do not specify a value for scale, the current scale setting remains in effect. The default scale value is 2.

If the scale is not required, you must set the scale to 0 (zero).

MAX

Sets the precision to the maximum allowable value for the indicated data type:

Data Type	MAX Precision
INTEGER	11
DECIMAL	18
REAL	9
FLOAT	20

Note: When issuing the CREATE SYNONYM command while the CONVERSION command is active in the profile, the Master File reflects the scale and length that is set by the CONVERSION command.

However, when issuing a SELECT statement, the answer set description does not use the information in the Master File. The length and scale used for the answer set description depends on whether a CONVERSION command is in effect.

If a CONVERSION command is in effect, the answer set description uses the length and scale that is set by the CONVERSION command.

If a CONVERSION command is not in effect, the answer set description uses the actual length and scale of the data.

Example: **Setting the Precision and Scale Attributes**

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to 7:

```
ENGINE ADAPTER_ID SET CONVERSION INTEGER PRECISION 7
```

The following example shows how to set the precision attribute for all DOUBLE PRECISION fields to 14 and the scale attribute to 3:

```
ENGINE ADAPTER_ID SET CONVERSION FLOAT PRECISION 14 3
```

The following example shows how to set the precision attribute for all INTEGER and SMALLINT fields to the default:

```
ENGINE ADAPTER_ID SET CONVERSION INTEGER RESET
```

The following example shows how to set the precision and scale attributes for all fields to the defaults:

```
ENGINE ADAPTER_ID SET CONVERSION RESET
```

Metadata Services With SQLENGINE SET

When the server is dedicated to accessing one Relational Database Management System (RDBMS) using the SET SQLENGINE command in the global profile, metadata calls to the server are processed against the native catalogs of the RDBMS.

How Applications Access Metadata

The metadata procedures used by applications to query the native catalog directly are:

- For an ODBC-enabled application, ODBC SQL calls.

- For an API-enabled application, EDARPC for ODBC*xxxx* calls.

The following table shows the relationship between the ODBC call and the API call:

ODBC Call	API Call
SQLTables	ODBCTABL
SQLColumns	ODBCCOLS
SQLPrimaryKeys	ODBCPKEY
SQLStatistics	ODBCSTAT
SQLProcedures	ODBCPROC
SQLProcedureColumns	ODBCPRCC
SQLSpecialColumns	ODBCSCOL
SQLColumnPrivileges	ODBCCPRV
SQLForeignKeys	ODBCFKEY
SQLTablePrivileges	ODBCTPRV

You can use the following two commands to control or override table and column metadata calls:

```
SQL sqlengine SET ODBCCOLSSORT
```

```
SQL sqlengine SET ODBCTABL
```

You can include these commands in any of the supported server profiles.

Obtaining Column Information (Db2 only)

When you issue either the SQLColumns or ODBCCOLS Metadata call from a client application, by default, the server requests the columns from Db2 in *colno* order.

Syntax: How to Request the Sort Order of Column Data

```
SET ODBCCOLSSORT {ON|OFF}
```

where:

ON

The column data is requested from the server with order by colno. This value is the default.

OFF

The column data is returned in unsorted order.

Obtaining User-Defined Metadata

When a client application issues either an ODBC or API metadata call, the server runs internal procedures that issue default SQL against the native RDBMS catalogs. You can issue your own SQL to run in place of the default SQL. You can specify this type of override for any of the internal server routines that deal with metadata.

Syntax: How to Request User-Defined Metadata

```
SQL sqlengine SET ODBCxxxx procname
```

where:

ODBC*xxxx*

Is the internal server routine name. Possible values are: ODBCTABL, ODBCCOLS, ODBCPKEY, ODBCSTAT, ODBCPROC, ODBCPRCC, ODBCSCOL, ODBCCPRV, ODBCFKEY, and ODBCTPRV.

procname

Is the procedure to run when the server receives the metadata call. This procedure must be available through the FOCEXEC ddname allocation in the server JCL.

Note: If this override procedure is used on a server running under MVS, it will add approximately 600K of storage above the line for each user.

When coding an override procedure, care must be taken to maintain the select list (answer set layout). The select list must conform to the ODBC specification for the return from the SQLTables call. See the *ODBC 2.0 Programmer's Reference and SDK Guide* for the SQLTables specification layout. Also, when using this override procedure, the parameters that are sent with the Metadata call need to be parsed correctly.

The override procedure should take the following format:

1. Dialogue Manager code to parse metadata call parameters.
2. `SQL sqlengine SELECT code;`

3.
```
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
```

Items 1 and 2 above are user coded; item 3 must always be present at the end of the procedure as coded above.

Example: Returning a List of Tables

The following sample code found in *qualif*.EDARPC.DATA(DB2ODBC1) returns a list of tables that the connected user is authorized to INSERT, UPDATE, DELETE, and SELECT. It is a sample of how to code a Db2 override procedure for ODBCTABL. It is one of several ways to code SQL to return an answer set for the ODBCTABL call. You can code any relevant query as long as the SELECT list is maintained.

In a server profile, issue SQL DB2 SET ODBCTABL DB2ODBC1, and then execute the following RPC request on the server:

```
ODBCTABL  ,<NULL>,,,,0,0,*
```

The following example matches the above ODBCTABL call:

```
-*
-* Dialogue Manager code to parse the ODBCTABL parameter list
-*
-DEFAULTS 1=' ',2='%',3='%'
-IF &2 NE '<NULL>' THEN GOTO LAB1;
-SET &2 = '%';
-LAB1
-IF &3 NE ' ' THEN GOTO LAB2;
-SET &3 = '%';
-LAB2
-*
-* SQL SELECT code
-*
SQL DB2
SELECT ' ',T2.CREATOR,T2.NAME,'TABLE',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE  = USER
AND   T1.TTNAME   LIKE '&2'
AND   T1.TCREATOR LIKE '&3'
AND   T2.TYPE     = 'T'
AND  (T1.DELETEAUTH IN  ('G','Y')
OR    T1.INSERTAUTH IN  ('G','Y')
OR    T1.SELECTAUTH IN  ('G','Y')
OR    T1.UPDATEAUTH IN  ('G','Y'))
AND   T1.TTNAME   = T2.NAME
AND   T1.TCREATOR = T2.CREATOR
UNION
```

```
SELECT ' ',T2.CREATOR,T2.NAME,'VIEW',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE  = USER
AND   T1.TTNAME   LIKE '&1'
AND   T1.TCREATOR LIKE '&2'
AND   T2.TYPE     = 'V'
AND  (T1.DELETEAUTH IN  ('G','Y')
OR    T1.INSERTAUTH IN  ('G','Y')
OR    T1.SELECTAUTH IN  ('G','Y')
OR    T1.UPDATEAUTH IN  ('G','Y'))
AND   T1.TTNAME   = T2.NAME
AND   T1.TCREATOR = T2.CREATOR
ORDER BY CREATOR,NAME;
-*
-* The following code must always be present
-*
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
```

Maintaining Upward Compatibility

In Version 4.3.x or earlier, an Extended Catalog (SYSOWNER table) was created at installation time. This provided additional control to the list of tables returned with the SQLTables or ODBCTABL call. In Version 5.1.0 and higher, this table is no longer created. If you need to use this table from a previous version of the server to continue to have control over the list of tables, issue the following in any supported server profile:

```
SQL sqlengine SET OWNERID ownerid
```

where:

sqlengine

Indicates the data source. You can omit this value if you previously issued the SET SQLENGINE command.

ownerid

Identifies the creator or owner of the Extended Catalog table SYSOWNER. If this command is used, the SQL generated to provide a list of tables will be limited by the owner names in the SYSOWNER table.

This command is only supported for customers who have configured a Relational Gateway in previous releases of the server and want to continue with the same configuration.

Additional Master File Attributes

These topics describe how to modify your server and communications configuration files.

The following attributes enable you to provide descriptive information about tables and columns.

`REMARKS`

Is an optional attribute for documenting tables.

`DESCRIPTION`

Is an optional attribute for documenting columns.

`TITLE`

Is an optional attribute for supplying an alternate column title on a report to replace the FIELDNAME value, which is normally used.

For the server, client tools using the API or ODBC will not use the TITLE attribute. The TITLE attribute is available only when directly querying the server catalog.

Documenting a Table

The REMARKS attribute enables you to document a table.

Syntax: How to Document a Table

The syntax is

```
REMARKS=text ,$
```

where:

text

Is one line of text. It can consist of a maximum of 78 characters. If the text contains a comma, you must enclose the text in single quotation marks.

The REMARKS attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example: Documenting an Oracle Table

The following example shows how to document the Oracle table SAMPLE.

```
FILE=SAMPLE,SUFFIX=SQLORA,REMARKS=This is a sample Oracle table.,$
```

Documenting a Column

The DESCRIPTION attribute enables you to provide a comment for a column in a table.

You can also add documentation to a column declaration—or a segment or table declaration—by typing a comment following the terminating dollar sign. You can even create an entire comment line by inserting a new line following a declaration and placing a dollar sign at the beginning of the line. For the server, a comment added in this manner will not be available to any client application.

If you are using DEFINE fields in a Master File, you must place the DESCRIPTION attribute on a line by itself. The semicolon after the DEFINE must appear on the same line as DESCRIPTION=.

Syntax: How to Document a Column

The syntax is

```
DESC[RIPTION]=text ,$
```

where:

text

Is one line of text. It can consist of a maximum of 44 characters.

If the text contains a comma, you must enclose the text in single quotation marks.

The DESCRIPTION attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Note: Whenever possible, place the description on the same line with the attributes FIELDNAME and ALIAS, to conserve space.

Example: Documenting a Column

The following example shows how to provide a description for the column UNITS. The single quotation marks are required because the description contains a comma.

```
FIELD=UNITS,ALIAS=QTY,FORMAT=I6,DESC='This is quantity sold, not
returned',$
```

Example: Documenting a DEFINE Field

The following example shows how to provide a description for the DEFINE field ITEMS_SOLD.

```
DEFINE ITEMS_SOLD/D8=ORDERED-INVENTORY
;DESC=DAMAGED ITEMS NOT INCLUDED,$
```

Supplying an Alternate Column Title

When you generate a report, each column title in the report defaults to the name of the column as it appears in the table. However, you can change the default column title by specifying the TITLE attribute.

For the server, client tools using the API or ODBC will not use the TITLE attribute. The TITLE attribute is available only when directly querying the server catalog, or when using WebFOCUS (Windows version).

You can override both the FIELDNAME and TITLE attributes with AS phrases in your report request. To override an existing TITLE attribute, use the SET TITLE command. To control whether the TITLE attribute is propagated in the Master File of a HOLD file, use the SET HOLDATTR command.

The TITLE attribute has no effect in a report if the column is used with a prefix operator such as AVE. You can supply an alternate column title for columns with prefix operators by using the AS phrase.

If you are using DEFINE fields in a Master File, you must place the TITLE attribute on a line by itself. The semicolon after the DEFINE must appear on the same line as TITLE=.

Syntax: How to Change the Default Column Title

The syntax is:

```
TITLE='text' ,$
TITLE='text,text,...' ,$
TITLE='text   /' ,$
```

where:

text

Is any string that consists of a maximum of 64 characters.

You can split the text across as many as five separate lines. Use single quotation marks to delimit the text, and commas to divide the text into separate lines on the report output.

You can include blanks at the end of an alternate column title by entering a slash (/) in the last position that will be blank, followed by a closing single quotation mark.

The TITLE attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example: Changing the Default Column Title

The following example shows how to replace the default column title, LNAME, with a title of Client Name.

```
FIELD=LNAME,ALIAS=LN,FORMAT=A15,TITLE='Client Name',$

Client Name
-----------
```

Example: Changing the Default Column Title to a Two-line Column Title

The following example shows how to replace the default column title, LNAME, with a two- line column title of Client Name.

```
FIELD=LNAME,ALIAS=LN,FORMAT=A15,TITLE='Client,Name',$

Client
Name
------
```

Example: Controlling the Underline for a Column Title

The following example shows how to control the length of the underline for an alternate column title.

```
FIELD=LNAME,ALIAS=LN,FORMAT=A15,TITLE='Client,Name     /',$

Client
Name
---------------
```

Example: Specifying a Column Title for a DEFINE Field

The following example shows how to replace the default column title for the DEFINE field, ITEMS_SOLD, with a two-line column title of Items Sold.

```
DEFINE ITEMS_SOLD/D8=ORDERED-INVENTORY;TITLE='Items,Sold',$

Items
Sold
-----
```

Specifying an Online Help Message

The HELPMESSAGE attribute enables you to specify an online help message for a field on a data entry screen. It is available only for FOCUS data maintenance applications.

When a Help Message Appears

Messages specified with the HELPMESSAGE attribute appear in the TYPE area of the CRTFORM when you:

- Enter a value for a database field that is invalid according to the criteria defined by the ACCEPT attribute.
- Enter a value for a database field that fails a VALIDATE test.
- Enter a value for a database field that causes a format error.
- Place the cursor in a data entry field and press a help key.

Regardless of the condition that triggers the display of the help message, the same message will appear.

Syntax: How to Specify a Help Message

The syntax is

```
HELP[MESSAGE]=text ,$
```

where:

text

Is one line of text. It can consist of a maximum of 78 characters.

You can use all characters and digits. If the text contains a comma, you must enclose the text in single quotation marks. Leading blanks are ignored.

The HELPMESSAGE attribute cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself.

Example: Displaying a Help Message to List Valid Values for a Column

The following example shows how to display a help message when the DEPARTMENT value is other than MIS, PRODUCTION, or SALES. DEPARTMENT has an ACCEPT attribute which tests values entered for it.

```
FIELDNAME=DEPARTMENT,ALIAS=DPT,FORMAT=A10,
   ACCEPT=MIS PRODUCTION SALES,
   HELPMESSAGE='DEPARTMENT MUST BE MIS, PRODUCTION, OR SALES',$
```

If you enter a value other than MIS, PRODUCTION, or SALES for DEPARTMENT on a CRTFORM, both the default message and help message display on the screen:

```
DATA VALUE IS NOT AMONG ACCEPTABLE VALUES FOR DEPARTMENT
DEPARTMENT MUST BE MIS, PRODUCTION, OR SALES
```

Example: Displaying a Help Message When a Format Error Occurs

The following example shows how to display a help message when a format error occurs in HIRE_DATE. Note that the format for HIRE_DATE is integer.

```
FIELDNAME=HIRE_DATE,ALIAS=HDT,FORMAT=I6YMD,
HELPMESSAGE=THE FORMAT FOR HIRE_DATE IS I6YMD,$
```

If you enter alphanumeric characters for HIRE_DATE on a CRTFORM, both the default message and help message display on the screen:

```
FORMAT ERROR IN VALUE ENTERED FOR FIELD HIRE_DATE
THE FORMAT FOR HIRE_DATE IS I6YMD
```

Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Setting*s from the menu. The Change Settings pane opens.

Syntax: How to Optimize Requests

```
SQL ADAPTER_ID SET {OPTIMIZATION|SQLJOIN} setting
```

where:

ADAPTER_ID

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

The adapter values are:

Adapter	Adapter ID	Adapter	Adapter ID
1010data	sql1010	Cache	dbsqlism
Db2	dbdb2	Excel	dbsqlexc
HP Vertica	dbsqlvrt	Hyperstage	dbsqlhyp
i Access	dbsqliia	CA-IDMS/SQL	dbsqlidm
Informix	dbsqlinf	Ingres	dbsqling
Interplex	dbsqlipx	JBoss Application Server	dbsqlmmx
JDBC	dbsqljdb	MariaDB	dbmariadb
Microsoft Access	dbsqlmac	Microsoft SQL Server	dbsqlmss
Microsoft SQL Server ODBC	dbmsodbc	MySQL	dbsqlmys
Netezza	dbsqlnez	Nucleus	dbsqlnuc
ODBC	dbsqlodb	Oracle	dbsqlora
Oracle TimesTen	dbsqlott	PostgreSQL	dbsqlpst
Progress	dbsqlpro	PSQL	dbsqlpsq
SAP Hana	dbsqlhan	SQLBase	dbsqlbas
Sybase	dbsqlsyb	Teradata	dbsqldbc
Transoft	dbsqltrn	UniData	dbsqlund
UniVerse	dbsqluv		

`SQLJOIN`

Is a synonym for OPTIMIZATION.

setting

Is the optimization setting. Valid values are as follows:

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some cases. However, misjoined unique segments and multiplied lines in PRINT-based and LIST-based report requests do not disable optimization. This is the default.

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

Both OPTIMIZATION settings produce the same report.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization. The report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000*n*, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
SQL ADAPTER_ID SET OPTIMIZATION OFF
 JOIN EMP_ID IN EMPINFO TO ALL WHO IN FUNDTRAN AS J1
 TABLE FILE EMPINFO
  SUM AVE.CURRENT_SALARY ED_HRS BY WHO BY LAST_NAME
  IF DEPARTMENT EQ 'MIS'
 END
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S):
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT
    FROM 'USER1'.'EMPINFO' T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM 'USER1'.'FUNDTRAN' T2 WHERE (T2.EID = ?)
    FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request. The server only formats the report.

```
SQL ADAPTER_ID SET OPTIMIZATION ON
 JOIN EMP_ID IN EMPINFO TO ALL WHO IN FUNDTRAN AS J1
 TABLE FILE EMPINFO
  SUM AVE.CURRENT_SALARY ED_HRS BY WHO BY LAST_NAME
  IF DEPARTMENT EQ 'MIS'
 END
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
    SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT)
      FROM 'USER1'.'EMPINFO' T1,'USER1'.'FUNDTRAN' T2
      WHERE (T2.EID = T1.EID) AND (T1.DPT = 'MIS')
      GROUP BY T2.EID,T1.LN
      ORDER BY T2.EID,T1.LN;
```

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests to Pass Virtual Fields Defined as Constants

A virtual field defined as a constant is passed directly to an SQL database engine (RDBMS) for optimized processing that takes advantage of RDBMS join, sort, and aggregation capabilities, thus reducing the volume of RDBMS-to-server communication and improving response time. (In prior releases, constants were not passed to the database engine, causing optimization to be turned off.)

Constants in the following formats are passed to the RDBMS: NUMERIC, CHAR, VARCHAR, CHAR/INTEGER combination, and DATE. (Note that a few formats are *not* passed to the RDBMS; these are CHAR and VARCHAR combination, TIME, and DATETIME.)

When valid constants are passed to an RDBMS engine, a report is calculated based on the defined value(s) and only the calculated subset of records is returned. The extent of data manipulation at the adapter level is limited, thereby improving performance.

Example: Passing Numeric Constants

```
DEFINE FILE SMIX87
INTEGERCONST/I4 = 5
REALCONST/D20.2 = -97995.38
NUMERICCONST/P13.4 = -92999.3647

TABLE FILE SMIX87
 SUM INTEGERCONST REALCONST NUMERICCONST MAX.QUOT.FA01INTEGER
 BY QUOT.FA02INTEGER
 END
```

The following SQL is generated:

```
SELECT T2."FA02INTEGER", SUM(5), SUM(-97995.38),
   SUM(-92999.3647), MAX(T2."FA01INTEGER") FROM TMIX83A T2 GROUP
   BY T2."FA02INTEGER" ORDER BY T2."FA02INTEGER";
```

Example: Passing a CHAR Constant

```
DEFINE FILE SMIX87
 CHARCONST/A10 = '2N'

TABLE FILE SMIX87
 SUM CHARCONST MIN.FA02CHAR_15
 BY QUOT.FA02INTEGER
 END
```

The following SQL is generated:

```
SELECT T2."FA02INTEGER", MAX('2N'), MIN(T3."FA02CHAR_15") FROM
   TMIX83A T2,TMIX86A T3 WHERE (T3."FA01INTEGER" =
   T2."FA01INTEGER") GROUP BY T2."FA02INTEGER" ORDER BY
   T2."FA02INTEGER";
```

Example: Passing CHAR and INTEGER Constants

```
DEFINE FILE SMIX87
 CHARCONST/A10 = '2N'
 INTEGERCONST/I4 = 5

TABLE FILE SMIX87
SUM CHARCONST INTEGERCONST
BY QUOT.FA02INTEGER
END
```

The following SQL is generated:

```
SELECT T2."FA02INTEGER", MAX('2N'), SUM(5) FROM TMIX83A T2
GROUP BY T2."FA02INTEGER" ORDER BY T2."FA02INTEGER";
```

Chapter 2

Using the Adapter for 1010data

The Adapter for 1010data allows applications to access specific 1010data databases. The adapter converts application requests into 1010data calls and returns optimized answer sets to the requesting application.

The adapter utilizes the ODBC API and is only available on Windows. The adapter is READ-ONLY. It does not support inserting data into 1010data tables in any way.

In this chapter:

Configuring the Adapter for 1010data

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Preparing the 1010data Environment

To use the Adapter for 1010data, you must first install the 1010data ODBC driver and all related libraries. To use this driver, you must configure a working data source via the ODBC Data Source Administrator. The Adapter will use this data source to access the 1010data database.

Configuring the Adapter

In order to connect to the 1010data database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).

- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one 1010data database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the 1010data Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for 1010data

The 1010data adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

This is the data source set in the ODBC Data Source Administrator.

Security

There are two methods by which a user can be authenticated when connecting to a data source:

- **Explicit.** The user ID and password are explicitly specified for each connection pass to the ODBC data source, at connection time, for authentication.
- **Trusted.** The adapter connects to the ODBC data source as a Windows login using the credentials of the operating system user impersonated by the server data access agent.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

File DSN parameters (optional)

Is the path to the ODBC File DSN that is defined on your Windows machine or network.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

For User/System DSN:

```
ENGINE SQL1010 SET CONNECTION_ATTRIBUTES connection
   DSNname/user_name,password,
```

For File DSN:

```
ENGINE SQL1010 SET CONNECTION_ATTRIBUTES connection
    FileDSN_name/,:'filedsn=FileDSN_path;',
```

where:

`SQL1010`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

DSN_*name*

Is the 1010data Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

user_name

Is the primary authorization ID by which you are known to an 1010data data source.

password

Is the password associated with the primary authorization ID.

FileDSN_*name*

Is the name of 1010data File Data Source Name (DSN) you wish to access.

FileDSN_*path*

Is the path to the 1010data File DSN, which may reside on the hard drive or on the network.

Example: Declaring Connection Attributes

For User/System DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the 1010data DSN named SAMPLESERVER.

```
ENGINE SQL1010 SET CONNECTION_ATTRIBUTES CON1
  SAMPLESERVER/SAMPLEUSER,SAMPLEPASSWORD,
```

For File DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 via the ODBC File DSN named SAMPLESERVER.dsn.

```
ENGINE SQL1010 SET CONNECTION_ATTRIBUTES CON2
SAMPLESERVER.dsn /,:'filedsn=C:\ SAMPLESERVER.dsn;'
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQL1010 SET DEFAULT_CONNECTION connection
```

where:

`SQL1010`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQL1010 SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQL1010 SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQL1010`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing 1010data Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the 1010data data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQL1010 to identify the Adapter for 1010data.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQL1010 [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQL1010

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each 1010data table or sheet that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.

- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for 1010data

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Note: Due to internal settings of the 1010data, filtering by Owner/Schema might not be applicable to some database objects.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.

- ❑ To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for 1010data synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File

```
FILENAME=ONHAND, SUFFIX=SQL1010, $
  SEGMENT=ONHAND, SEGTYPE=S0, $
    FIELDNAME=PROD_NUM, ALIAS=Prod_Num, USAGE=A255V, ACTUAL=A255V,
      MISSING=ON, $
    FIELDNAME=PRODNAME, ALIAS=Prodname, USAGE=A255V, ACTUAL=A255V,
      MISSING=ON, $
    FIELDNAME=QTY_IN_STOCK, ALIAS=Qty_in_Stock, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=PRICE, ALIAS=Price, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=COST, ALIAS=Cost, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
```

Access File

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File for 1010data.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the 1010data table. The value assigned to this attribute can include the name of the workbook as follows: TABLENAME=[*filename*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Customizing the 1010data Environment

The Adapter for 1010data provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to 1010data.

Syntax: **How to Issue the TIMEOUT Command**

```
ENGINE SQL1010 SET TIMEOUT {nn|0}
```

where:

SQL1010

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

0

Represents an infinite period to wait for a response.

1010data Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 3

Using the Adapter for Adabas

The Adapter for Adabas allows applications to access Adabas data sources. The adapter converts application requests into native Adabas statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Adabas Environment

Adabas environment variables must be set before you configure the server. Refer to the Software AG documentation for more information about required environment variables.

Procedure: How to Prepare the Adabas Environment on Windows

On Windows, the Adabas environment is set up during the installation of Adabas.

Procedure: How to Prepare the Adabas Environment on UNIX

1. Specify the Adabas database directories to access using the UNIX environment variable $ACLDIR . For example:

```
ACLDIR =/rdbms/sag/acl
export ACLDIR
```

2. Specify the Adabas driver using the UNIX environment variable $ADALNK. For example:

```
ADALNK=/rdbms/sag/ada/v6xxx/adalnkx.so
export ADALNK
```

3. Specify the version of the Adabas product using the UNIX environment variable $ADAVERS. For example:

```
ADAVERS=v6xxx
export ADAVERS
```

Procedure: How to Prepare the Adabas Environment on OpenVMS

When a site creates an Adabas ID, the software automatically generates a DCL setup script so users can properly invoke the environment.

The specification for the location of the Adabas setup file is

```
$@SAG$ROOT:[adabas_root]LOGIN.COM
```

where:

`SAG$ROOT`

Is the disk on which Adabas is installed.

`adabas_root`

Is the root directory of the Adabas installation.

Example: Specifying the Adapter for Adabas on OpenVMS

```
$@SAG$ROOT:[Adabas]LOGIN.COM
```

The logicals ADABAS$MAIN, ADALNK, and ADABAS$VERSION will be used to communicate with Adabas Nucleus.

Procedure: How to Prepare the Adabas Environment on z/OS

Before starting the server, you can specify the name of the CEE LKED PDS by exporting the environment variable CEELKED. The default value is the standard IBM name, CEE.SCEELKED. If you wish to override the default, define the variable as follows

```
CEELKED=cee_lked_pds
```

on the EDAENV member of the *high_level_qualifier.server_type*.DATA.

The CELLKED variable value (either the default or a value you specify) is displayed along with other environment parameters, such as STEPLIB and LIBPATH, in the lower section of the Adabas Configuration pane on the Web Console.

Configuring the Adapter for Adabas

Configuring the adapter consists of specifying connection information for each of the connections you want to establish.

No user ID and password are needed to establish a connection to Adabas. This means that Trusted Mode is the standard for Adabas. To secure a connection, use the SET PASSWORD feature.

Procedure: How to Configure the Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Note:

- On z/OS platforms, the adapter needs to be configured before connections can be tested.
- More than one connection can be configured. To add connections, right-click the *Adabas* folder, choose *Add Connection* and repeat Steps 5 and 6.

Reference: Connection Attributes for Adabas

The Adabas adapter is under the *DBMS* group folder.

This following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection Name

Is the logical name used to identify this particular set of connection attributes. There is no default Connection Name. You must provide a value.

The Connection Name is a new parameter added in Version 7 Release 7.4. It is designed to support access to different databases with different SVCs and releases through the same adapter. It corresponds to the new CONNECTION parameter in an Access File.

The adapter continues to support connection strings and Access Files without a Connection Name parameter, as they were declared prior to Version 7 Release 7.4.

Note that the connection name <*local*> will be displayed on the Web Console for the legacy connection strings.

Users may assign a more meaningful name instead. For example, the legacy connection

```
ENGINE ADBSINX SET CONNECTION_ATTRIBUTES
/,;3:',ADAEMPL=1,ADASVC=214,ADADEVICE=3,390,ADAASSO=SAGLIB.ADA814.
IBIMVS.DB03.ASSOR1'
```

can be converted into new connection supported since Version 7 Release 7.4

```
ENGINE ADBSINX SET CONNECTION_ATTRIBUTES
ADA814/,;3:',ADAEMPL=1,ADASVC=214,ADADEVICE=3,390,ADAASSO=SAGLIB.ADA814.IBIMVS.DB03
.ASSOR1'
```

Default DB number

Number of the database to access. For example: 001(Required for all platforms.)

SAG Employee/Demo file number

Number of the Adabas Employee or another Demo File. For example: 011 (Required for all platforms.)

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

SVC number

SVC number of the Adabas router. For example: 240 (Required only for z/OS.)

Device type

Type of device where the Adabas Associator is located. For example: 3390 (Required only for z/OS.)

ADABAS Load Library name

Name of the Adabas load library that contains the zapped member ADALNKR (see notes below). This name needs to be added after the IBI load library; the order is important. For example:

```
//TASKLIB ...
   // DD DISP=SHR,DSN=IBI.HOME.LOAD
   // DD DISP=SHR,DSN=SAGLIB.ADA814.LOAD
   ...
```

(Required only for z/OS.)

Associator data set name

Name of the Adabas Associator data set. For example: SAGLIB.ADA814.EXAMPLE.DB001.ASSOR1 (Required only for z/OS.)

ADABAS LOAD library

This parameter is needed only if the Adapter will be used to access Adabas databases with different SVCs. It lets you specify your own high_level_qualifier for each database.

If only one SVC is used, this parameter is not needed.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the *Profile Name* field (The extension is added automatically).

Store the connection attributes in the server profile (edasprof).</p>

Note:

- The Adabas adapter configuration procedure was changed starting with the Version 7 Release 7 of the server. This procedure no longer uses SVC, since starting with Adabas 8, Software AG no longer provides a source for ADALNK. The ADALNKR needs to be prepared as described in the Software AG Adabas 8 documentation. See *Installing Adabas with TP Monitors* under the *Installing Adabas with Batch/TSO under Adabas 8* chapter. The Default SVC number needs to be zapped into ADALNKR prior to the Adabas adapter configuration step.
- The correct SVC number still needs to be provided on the configuration screen. It is used by adapter to create synonyms and in other routines.

Overriding Default Passwords in Specific Files

You can set passwords for Adabas files in the server profile using the SET PASSWORD command. You can set default passwords for all files and/or databases. Specific passwords can be set for specific files which will override the default. The SET command overrides the password coded in the Access File. The SET PASSWORD command remains valid throughout the user session.

Syntax: How to Use the SET PASSWORD Command

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET PASSWORD password FILENO ALL DBNO {ALL|dbno}
ENGINE ADBSINX SET PASSWORD password FILENO fileno DBNO dbno
ENGINE ADBSINX SET PASSWORD OFF
ENGINE ADBSINX SET PASSWORD DEFAULT
```

where:

`ADBSINX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

password

Is the password, which can be from one to eight characters in length.

FILENO

Specifies the file number for which the password is set.

DBNO

Specifies the database or databases for which the password is set.

ALL

Indicates all files and/or databases used with the FILENO and DBNO parameters. If you want to use ALL for both FILENO and DBNO, issue this command before any other subsequent password commands. ALL overrides any prior settings.

dbno

Is any valid numeric database value between 0 and 255 (65,535 is the maximum value on a mainframe platform). The DBNO parameter indicates the actual database number.

fileno

Provides a file number, a list of file numbers, and/or a range of file numbers used with the FILENO parameter. Numbers and ranges can be combined by separating items with commas. Valid file numbers range between 0 and 255 (5000 is the maximum value on a mainframe platform).

OFF

Clears all previous ADBSINX SET PASSWORD commands. The Adapter for Adabas does not use any passwords specified in the Access File. This command lets you access only those files that have no password security.

DEFAULT

Clears all previous ADBSINX SET PASSWORD commands and causes the Adapter for Adabas to use the password in the Access File.

To set a default password:

- Issue the SET PASSWORD command using the ALL keyword for FILENO and/or DBNO.
- Issue specific password requests by specifying particular file and/or database numbers (or ranges) in a subsequent SET PASSWORD command.

Note that subsequent requests are appended to previous requests. Changes are cumulative. The passwords can be issued from a FOCEXEC, which may be encrypted.

Example: Using the SET PASSWORD Command to Specify Files in a Database

To set the password to BRUCE for specific files in database number 123 on z/OS, issue the following command:

```
ENGINE ADBSINX SET PASSWORD BRUCE FILENO 1,3-5,23,89-93 DBNO 123
```

To clear all previous ADBSINX SET PASSWORD commands on z/OS, and tell Adabas to use the password in the Access File, issue the following command:

```
ENGINE ADBSINX SET PASSWORD DEFAULT
```

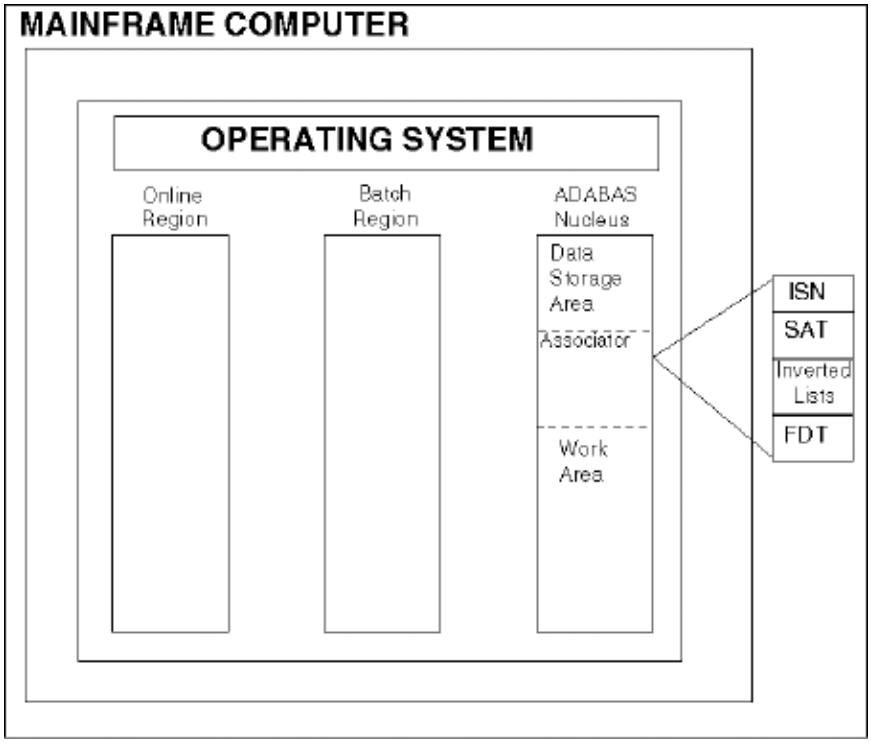
Adabas Overview

This topic provides an overview of Adabas and Adabas files, and includes a discussion of server concepts.

Adabas is a field-oriented DBMS. Data retrievals and updates are performed on a field-by-field basis.

Since Adabas data retrieval occurs at the field level rather than at the record level, your applications may be designed without consideration for the physical organization and maintenance of the record. Data is accessed in a variety of ways (in physical sequence, in logical sequence, or by Internal Sequence Numbers), thereby enabling you to tailor data access to address your specific needs.

The Adabas DBMS consists of several components. The following is an illustration of the Adabas environment.



This table describes the components of the Adabas environment.

Component	Description
Operating System	Set of programs (software) that control the operation of the computer (hardware).
Online Region	Part of the Dynamic Area of the computer. It is the section in which multiple jobs execute simultaneously. The Online Region is also known as the Foreground Region (applies only to TSO).
Batch Region	Part of the Dynamic Area of the computer. Jobs become a series of commands that are grouped together and processed in batches (applies only to batch jobs).

Component	Description
Data Storage Area	Contains the actual data. The data is stored in compressed form.
Associator	Stores relationships about the data. It contains the following components: ❏ Internal Sequence Number (ISN) list, also called the Address Converter, used to determine the physical location of data. ❏ Space Allocation Tables (SATs) to control storage space. ❏ Inverted lists for the defined descriptors. Inverted lists and descriptors are discussed in *Inverted Lists: The Adabas Key Structure* on page 139. ❏ Field Definition Tables (FDTs), which contain detailed data structure information for each field and descriptor.
Work Area	Is a "scratch pad" used by Adabas to build ISN lists and to sort and store records that your program requires.

The Multi-Programming Module (MPM), the cornerstone of the Adabas Nucleus, contains the logic that enables multiple programs, both batch and online, to access Adabas databases simultaneously.

Each database has its own data storage area, associator, and work area, in addition to its own unique database number.

Adabas Files

When accessing Adabas, the server uses logical files as the unit of data retrieval. An Adabas file is identified by a file number. The number is unique within each database.

All file and field documentation for the Adabas database can be stored in the Predict dictionary. The information about the data structure includes processing uses, file ownership, and field descriptions.

Inverted Lists: The Adabas Key Structure

An Adabas file is defined with a set of indexes called descriptors. The Adabas term for these values, which reside in the File Associator Table, is inverted lists. Inverted lists contain the values of descriptors, a count of the total number of records in which each value appears, and the Internal Sequence Numbers (ISNs), in ascending order, associated with each occurrence.

Adabas utilities create and maintain an inverted list for each descriptor identified. These lists store data in an ascending sequence by the value of the key. A single file may have up to 256 inverted lists associated with it.

The descriptors are identified to speed data location and to retrieve specific, frequently required data from Adabas files. Of the available types of inverted lists, the Adapter for Adabas supports the following.

Inverted List	Description
Descriptors	Key values associated with a single field (also called elementary field descriptors).
Subdescriptors	Key values associated with part of a single field.
Superdescriptors	Key values associated with all or part of two to twenty fields.
Phonetic descriptors	Alphabetic values associated with the first twenty bytes of the field value.
Hyperdescriptors	A value generated, based on a user-supplied algorithm.

In the following text, descriptor refers to all five types of descriptors interchangeably, unless otherwise indicated.

Two or more files may be related in these ways:

- The database administrator may couple several files based on a common descriptor key in each file.
- An application program may logically relate files if there are fields in one file that can be used to access a key in another.

Field Definition Tables

Field description information for Adabas files is maintained in the FDT, which is part of the Adabas associator. Adabas uses this information when accessing files. The FDT includes the following information:

Field Information	Description
Field Level Indicator	Denotes a hierarchical relationship between fields. In an Adabas record description the levels are 1, 2, 3, and so on.
Adabas Field Name	Two-character name used by Adabas to identify the field. This name is unique to the file. The first character must be alphabetic, and the second character can be alphabetic or numeric. Field names E0 through E9 are reserved for internal Adabas use.
Standard Length	Length of the field in bytes.
Standard Format	Format of the field. Refer to the table in *Standard Formats for Adabas Fields* on page 141 for the acceptable formats and their meanings.
Field Properties	Indicates whether the field value is null suppressed (NU), the field is of fixed or variable length (FI), or if the field (which exists on a mainframe platform) is long alphanumeric (LA).
Descriptor Status	Indicates if the field is a descriptor, subdescriptor, superdescriptor, phonetic descriptor, hyperdescriptor, subfield, or superfield.
Field Type	Indicates if a field is a group (GR), multi-value (MU), or periodic group (PE). Multi-value fields and periodic groups are examples of multiply occurring fields.

Typically, an Adabas FDT contains some, but not all, of the field characteristics discussed above. To view the FDT, Software AG provides the ADAREP report. See your Software AG documentation for more information on how to run the ADAREP report.

The external field name is in the Predict dictionary.

External Field Name	32-byte name used in a Predict to identify the field. This name is unique to the file.

Reference: **Standard Formats for Adabas Fields**

The following table shows the acceptable standard formats for Adabas fields and their meanings.

Format	Description
A	Alphanumeric data field with a maximum of 253 bytes (126 for descriptor fields).
U	Zoned decimal data field with a maximum of 29 bytes (signed, unpacked data).
P	Signed packed decimal data field with a maximum of 15 bytes.
B	Unsigned binary data field with a maximum of 126 bytes.
F	Single-precision, floating point data field that is always four bytes long.
G	Decimal, double-precision integer field with an eight-byte maximum.

In the following example, the column on the left illustrates a partial Adabas FDT. The bold numbers on the left refer to the numbered annotations that follow:

```
   PAY-FILE

   FIELD DESCRIPTION (from FDT)          FIELD NAME (from DDM)

1. 01, PS, 08, P, NU, DE                 SSN
2. 01, PW, 04, P                         HOURLY_WAGE
3. 01, MI, PE                            MONTHLY_INFO
   02, MH, 04, P                         MONTHLY_HOURS
   02, MW, 04, P                         MONTHLY_WAGES
   02, MT, 04, P                         MONTHLY_TAX
4. 01, PT, 04, P, DE, MU                 TAX
   01, PC, 08, P, DE, MU                 CHILD_SSN
```

1. The first field, SSN, is an elementary-level field. Its internal name is PS, it is eight bytes long, and it is in packed decimal data format. The field is null suppressed, as indicated by the NU, and it is a descriptor (DE).

2. HOURLY_WAGE is also an elementary-level field. It is called PW internally, is four bytes long, and is also in packed decimal data format.
3. MONTHLY_HOURS, MONTHLY_WAGES, and MONTHLY_TAX are all subordinate fields of the MONTHLY_INFO periodic group (PE). Internally, they are called MH, MW, and MT, respectively. Each is four bytes long and is in packed decimal data format.
4. TAX and CHILD_SSN are elementary-level fields. TAX is called PT internally, is four bytes long, is in packed decimal data format, and is a multi-value (MU) field with a descriptor (DE) associated with it. CHILD_SSN is called PC internally, is eight bytes long, is in packed decimal format, and is also a multi-value field with a descriptor associated with it.

For more information about Adabas periodic groups (PE) and multi-value (MU) fields, see *Mapping Adabas Files With Variable-Length Records and Repeating Fields* on page 192.

Managing Data Storage With Null-Suppression

Null-suppression is one of the methods Adabas employs to manage data storage. Only the fields within a record that contain data are stored. Fields containing blanks for alphabetic characters or zeros for numeric data are not stored. They are represented by a one-byte "empty field" character. When a procedure accesses a record containing a null-suppressed field, Adabas expands the field to its full length and includes the null values of blanks or zeros.

When the null-suppressed field is a descriptor or part of a superdescriptor, no entry is made for the record containing that field in the inverted list. It would not be productive to have the inverted list direct you to records in which the descriptor has no data.

A field with null-suppression appears with the NU attribute in the Adabas FDT.

Note that for the Server to support NULL (MISSING = ON) for an Adabas field, you must define the field in the Adabas FDT with the NC attribute.

Managing Data Storage With SQL Null Options

Adabas includes two data definition options, Not Counted (NC) and Not Null or Null Value Not Allowed (NN), for providing SQL-compatible null representation for mainframe Adabas SQL Server (ESQ) by Software AG and other Structured Query Language (SQL) database query languages.

The NC and NN options cannot be applied to fields defined:

- With Adabas null suppression (NU).
- With a fixed-point data type (FI).
- With multiple-values (MU).
- Within a periodic group.

- As group fields.

NC: SQL Null Value Option

Without the Not Counted (NC) option, a null value is either zero or blank depending on the field format.

With the NC option, zeros or blanks specified in the record buffer are interpreted according to the "null indicator" value: either as true zeros or blanks (that is, as "significant" nulls) or as undefined values (that is, as true SQL or "insignificant" nulls).

If the field defined with the NC option has no value specified in the record buffer, the field value is always treated as an SQL null.

Note: On the mainframe platform, subdescriptors and superdescriptors defined with NC=YES and MISSING=ON parameters cannot be used to search for the SQL NULL value, as that causes an Adabas RC 61.

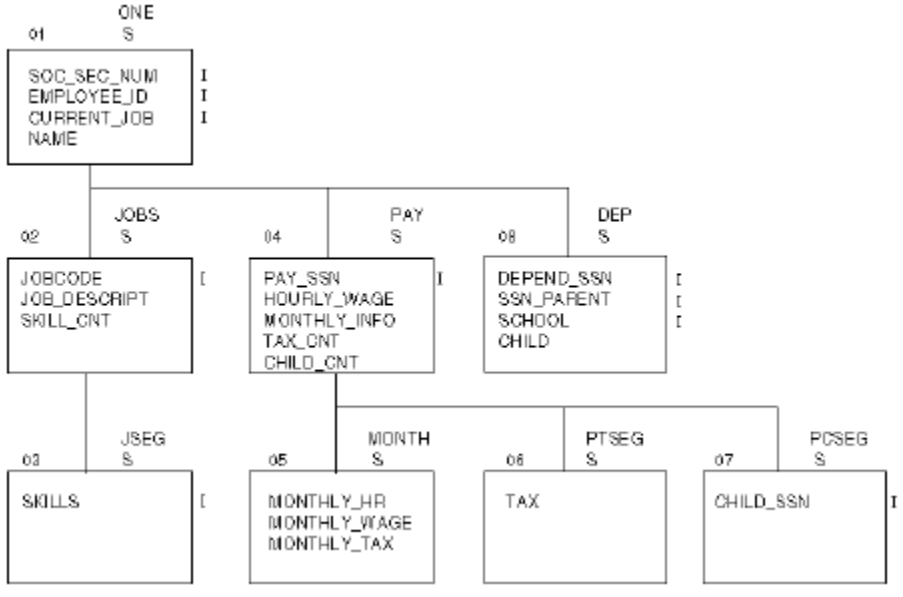
NN: SQL Not Null Option

The Not Null or Null Value Not Allowed (NN) option may only be specified when the NC option is also specified for a data field. The NN option indicates that an NC field must always have a value (including zero or blank) defined; it cannot contain "no value".

Server File Structure

The server uses a non-procedural language to create reports, graphs, and extract files. It also enables you to access data sources without knowing the details of the file structure or access method.

The server treats any data source as either a single-path or multi-path hierarchy. Graphically, information is laid out using an inverted tree structure, as in the following sample STAFF file. In the example, the letter I to the right of a field indicates an indexed field.



The most general information appears at the top, and the more specific information appears under it. Each box in the structure is referred to as a segment. A database can consist of one or more logically related segments.

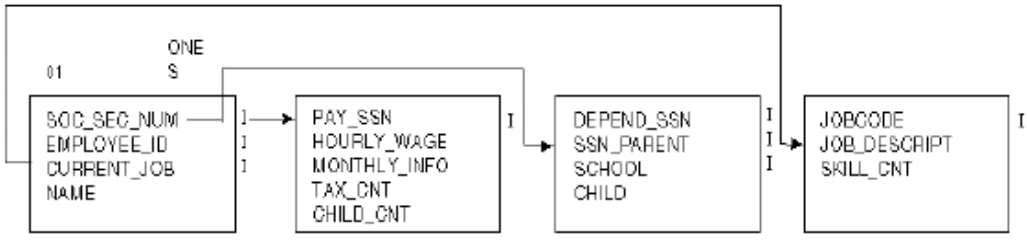
When logically related information is retrieved using this database structure, multiple occurrences of each segment are created. Each occurrence of a segment is called a segment instance. Each Adabas database is equivalent to a collection of logically related segment instances.

The retrieval sequence of the segments is determined by the view of the structure, that is, the order of the segments from top to bottom and left to right. The segment at the top is the parent, or root, segment. The segments under the parent are the child, or descendant, segments.

Generally, parent and child segments have a one-to-many relationship. That is, a single parent has multiple occurrences of one child segment. However, the server also handles one parent-one instance only of a child segment.

Adabas uses specific concepts and techniques for organizing data. It holds data in logically distinct files that are interrelated using fields that share common formats and values called descriptors.

Within a single Adabas file, records that contain multiple occurrences of a field can vary in length and format. The following example illustrates an Adabas structure containing four separate files that are linked by common fields:



Many Adabas structures are more complex than the preceding example.

Adabas Descriptors

Descriptors are fields, partial fields, or groups of complete and partial fields used by Adabas to select records in a file. Adabas descriptors correspond to indexed fields.

There are five types of descriptors supported by the Adapter for Adabas:

- Descriptors (DE), which have a value associated with a single field.
- Superdescriptors (SPR), which have a key value associated with all or part of two to twenty fields (see your Software AG documentation for current limitations on superdescriptors).
- Subdescriptors (NOP), which have a key value associated with part of a single field.
- Phonetic Descriptors (PDS), which have a similar phonetic value associated with a single field. The phonetic value of a descriptor is based on the first 20 bytes of the field value. Only alphabetic values are considered: numeric values, special characters, and blanks are ignored. Lowercase and uppercase alphanumeric characters are internally identical.
- Hyperdescriptors (HDS), which have a value generated, based on a user-supplied algorithm.

All five types of descriptors are collectively referred to as "descriptors."

Consider the Adabas FDT for the STAFF file:

```
STAFF
FILE FDT
    FIELD DESCRIPTION (from FDT)          FIELD NAME (from Predict)
       01, ES, 08, P, NU, DE              SOC_SEC_NUM
       01, EI, 06, A, NU, DE              EMPLOYEE_ID
       01, EJ, 03, A, DE                  CURRENT_JOB
       01, EN, 24, A                      NAME
```

Notice that SOC_SEC_NUM, EMPLOYEE_ID, and CURRENT_JOB are labeled as descriptors. Any one of these three fields can be used to search the STAFF file.

Adabas descriptors, superdescriptors, and subdescriptors must be declared in Master and Access Files in certain ways. For more information, see *Mapping Adabas Descriptors* on page 184.

Adabas Files With Fixed-Length Records

Unless an Adabas file has multi-value fields or periodic groups, each field occurs once in each record. When describing an Adabas file, you do not need to describe all the fields in your Master File; just the fields you use. Note that if you choose a periodic group, all fields in the periodic group must be defined. A single simple Adabas file maps to a single segment on the server, and each field you use in the Adabas file becomes a field in the segment.

Adabas RECORD		SEGMENT	
ES	SOC_SEC_NUM	SOC_SEC_NUM	ES
EI	EMPLOYEE_ID	EMPLOYEE_ID	EI
EJ	CURRENT_JOB	CURRENT_JOB	EJ
EN	NAME	NAME	EN

Adabas Files With Variable-Length Records and Repeating Fields

There are two types of Adabas repeating fields:

- Multi-value (MU) fields.
- Periodic (PE) groups.

An MU field is a single field that occurs a variable number of times in a record. It appears as type MU in an Adabas record description. Adabas supports up to 191 occurrences of MU fields per record.

A PE group is a group of contiguous fields that occur a variable number of times in a single record. It appears as type PE in an Adabas record description. Adabas supports up to 191 occurrences of PE fields per record. These component fields are regular fields or MU fields.

Note: The Adapter for Adabas supports the maximum number of occurrences of MU fields and PE groups allowed by Software AG.

When an Adabas file contains MU fields or PE groups, the repeating information occurs a variable number of times. The physical record length varies depending on how many times the repeating information actually appears.

More than one segment is needed to accurately describe an Adabas file with repeating data. The number of times the fields repeat is expressed in a counter field. For more information, see *Mapping Adabas Files With Variable-Length Records and Repeating Fields* on page 192.

Managing Adabas Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Adabas data types.

For the Adapter for Adabas to access Adabas files, you must describe each file you use in a Master and Access File. The logical description of an Adabas file is stored in a Master File, which describes the field layout.

Creating Synonyms

Synonyms define unique names (or aliases) for each Adabas table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server metadata.

The Create Synonym facility automatically generates Master and Access Files for Adabas files based on information stored in the Field Definition Table (FDT) in Adabas and the Predict dictionary (optionally). The command requires Adabas Release 5.0 or higher and Predict Release 3.1.4 or higher.

The syntax is identical on UNIX, Windows, and z/OS, and OpenVMS platforms.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Adabas

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Current DBID

Enter identification number of the database to access. (Required for all platforms.)

Associator data set name

Enter name of the Adabas Associator data set. (Required only for z/OS.)

Use Predict Metadata

Put a check mark in this checkbox to use metadata from an existing Predict file dictionary. This refreshes the pane, adding two entry boxes in which you can set the Default Predict Setting by entering the following:

`File Number`

Adabas file number where the Predict file is located.

`DBID`

Database ID (DBNO) for the Adabas DBMS containing the Predict dictionary data.

Then press *Select Tables* to summon the second Create Synonym pane (Step 2 of 2).

Provide values for the following superdescriptor processing parameters:

Mode

There are two options:

`Extended (NEW)`

Describes superdescriptors as groups in the Master File. Neither the groups nor their components (fields) have field names specified in the Master File. Aliases are populated using the Adabas FDT (2-byte name).

The Adapter for Adabas uses new logic to process the selection criteria of requests involving that synonym. Screening conditions within the request are analyzed and the superdescriptor that covers the highest order component fields required by the selection criteria are used. For more information, see *NEW Synonym Setting for Superdescriptors* on page 203.

Note that you must specify CALLTYPE=RL in the Access File in order to take advantage of superdescriptor-based access.

`Standard`

Incorporates the usual CREATE SYNONYM behavior, which does not trigger the new logic for requests using superdescriptors. Standard is the default value.

ISN field

No - Indicates that Adabas Internal Sequence Numbers (ISN) are not to be included in the Master File.

GFBID field

Global Format Buffer ID (GFBID) support is applicable only if a GFBID field is present in the Master File. This field has a user-defined field name, and an ALIAS of GFBID. This field is used to determine the Global Format Buffer ID that will be defined in read requests with identical field lists for the same database.

`Yes`

Specifies that a field generated in the Master File is to be used to specify Adabas Global Format Buffer ID values.

`No`

Indicates that Global Format Buffer ID values are not to be included in the Master File.

Enter the following additional parameters for synonym field name processing options:

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Adabas-specific Create Synonym Notes

- Only one Adabas file description can be generated in the Master File. If multiple Adabas files must be described in one Master File, a description for each Adabas file must be created and combined in one Master File manually.
- The Create Synonym facility describes Adabas files within the constraints of the Adapter for Adabas. For more information, see *Platform-Specific Functionality* on page 200. Superdescriptors, subdescriptors, periodic elements (PE groups), and multi-value fields (MU) are included. Comments are used to describe unsupported fields in the generated Master File.
- The Create Synonym facility allows you to customize the output using data from the Predict dictionary. You can:
  - Replace generic field names constructed based on Adabas field names (xx_FIELD) with the names obtained from the Predict dictionary.
  - Replace the USAGE format and length generated from an Adabas FDT (Field Definition Table) with the format and length from the Predict dictionary.

- Include the NATURAL column heading stored in the Predict dictionary in the Master File for use as the default column heading in server reports.

Example: **Creating Synonyms When Adabas Fields Are Defined With the NC Option**

The Create Synonym facility adds the parameter NC=YES in the Access File for fields defined with the NC option.

If a field with the NC option *does not have* the NN option, then the parameter MISSING=ON is also added to the field definition in the Master File. Subdescriptors and superdescriptors derived from this field will be defined as the fields with the following options: TYPE=NOP, NC=YES, and MISSING=ON. No component fields will be associated with these subdescriptors and superdescriptors.

Sample Master File and Access File contents with the NC option:

Master File:

```
FILENAME=ADANC1, SUFFIX=ADBSINX , $
  SEGMENT=S01, SEGTYPE=S0, $
    FIELDNAME=AA_FIELD, ALIAS=AA, USAGE=I6, ACTUAL=I2, FIELDTYPE=I, $
    FIELDNAME=AB_FIELD, ALIAS=AB, USAGE=I6, ACTUAL=I2, FIELDTYPE=I, MISSING=ON, $
    FIELDNAME=AC_FIELD, ALIAS=AC, USAGE=I6, ACTUAL=I2, FIELDTYPE=I, $
    FIELDNAME=AD_FIELD, ALIAS=AD, USAGE=I6, ACTUAL=I2, MISSING=ON, $
    FIELDNAME=AE_FIELD, ALIAS=AE, USAGE=I6, ACTUAL=I2, $
    FIELDNAME=BA_FIELD, ALIAS=BA, USAGE=A2, ACTUAL=A2, FIELDTYPE=I, $
    FIELDNAME=BB_FIELD, ALIAS=BB, USAGE=A2, ACTUAL=A2, FIELDTYPE=I, MISSING=ON, $
    FIELDNAME=BC_FIELD, ALIAS=BC, USAGE=A2, ACTUAL=A2, FIELDTYPE=I, $
    FIELDNAME=BD_FIELD, ALIAS=BD, USAGE=A2, ACTUAL=A2, MISSING=ON, $
    FIELDNAME=BE_FIELD, ALIAS=BE, USAGE=A2, ACTUAL=A2, $
   GROUP=G1_GROUP, ALIAS=G1, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=CA_FIELD, ALIAS=CA, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=CB_FIELD, ALIAS=CB, USAGE=A2, ACTUAL=A2, MISSING=ON, $
    FIELDNAME=CC_FIELD, ALIAS=CC, USAGE=A2, ACTUAL=A2, $
$ $$$$$$$$$$$$$$$$$$$$$ Superdescriptor $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $
    FIELDNAME=S1_FIELD, ALIAS=S1, USAGE=A6, ACTUAL=A6, FIELDTYPE=I, MISSING=ON, $
  $ FIELD=BA_FIELD_S01       ,ALIAS=BA         ,A2    ,A2    ,INDEX=I,$
  $ FIELD=BB_FIELD_S01       ,ALIAS=BB         ,A2    ,A2    ,INDEX=I,$
  $ FIELD=BE_FIELD_S01       ,ALIAS=BE         ,A2    ,A2    ,       ,$
$ $$$$$$$$$$$$$$$$$$$$$ Superdescriptor $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $
     FIELDNAME=S2_FIELD, ALIAS=S2, USAGE=A6, ACTUAL=A6, FIELDTYPE=I, MISSING=ON, $
$ FIELD=BA_FIELD_S02       ,ALIAS=BA         ,A2    ,A2    ,INDEX=I,$
$ FIELD=BC_FIELD_S02       ,ALIAS=BC         ,A2    ,A2    ,INDEX=I,$
$ FIELD=BD_FIELD_S02       ,ALIAS=BD         ,A2    ,A2    ,       ,$
$ $$$$$$$$$$$$$$$$$$$$$$$$ Subdescriptor $$$$$$$$$$$$$$$$$$$$$$$$$$$$ $
     FIELDNAME=S3_FIELD, ALIAS=S3, USAGE=A1, ACTUAL=A1, FIELDTYPE=I, MISSING=ON, $
$ $$$$$$$$$$$$$$$$$$$$$$$$ Subdescriptor $$$$$$$$$$$$$$$$$$$$$$$$$$$$ $
    FIELDNAME=S4_FIELD, ALIAS=S4, USAGE=A1, ACTUAL=A1, FIELDTYPE=I, $
$
```

Access File:

```
 RELEASE=6, OPEN=YES, $
 SEGNAM=S01, ACCESS=ADBS, FILENO=120, DBNO=3, CALLTYPE=FIND,
   UNQKEYNAME=AA_FIELD, WRITE=YES, $
$   CALLTYPE=RL  ,SEQFIELD=AA_FIELD                        ,$
$ FIELD=AA_FIELD ,TYPE=DSC       ,NC=   ,$
  FIELD=AB_FIELD, TYPE=DSC, NC=YES, $
  FIELD=AC_FIELD, TYPE=DSC, NC=YES, $
  FIELD=AD_FIELD, TYPE=, NC=YES, $
  FIELD=AE_FIELD, TYPE=, NC=YES, $
$ FIELD=BA_FIELD ,TYPE=DSC       ,NC=   ,$
  FIELD=BB_FIELD, TYPE=DSC, NC=YES, $
  FIELD=BC_FIELD, TYPE=DSC, NC=YES, $
  FIELD=BD_FIELD, TYPE=, NC=YES, $
  FIELD=BE_FIELD, TYPE=, NC=YES, $
  FIELD=CB_FIELD, TYPE=, NC=YES, $
  FIELD=CC_FIELD, TYPE=, NC=YES, $
  FIELD=S1_FIELD, TYPE=NOP, NC=YES, $
  FIELD=S2_FIELD, TYPE=NOP, NC=YES, $
  FIELD=S3_FIELD, TYPE=NOP, NC=YES, $
  FIELD=S4_FIELD, TYPE=NOP, NC=YES, $
```

Note: The following processing rules apply for fields defined with the NC=YES and MISSING=ON parameters:

- The default value is set to the SQL NULL value when a new record is created:

```
SQL
INSERT INTO ADANC1
   (AA_FIELD,   AB_FIELD,   AC_FIELD,
    BA_FIELD,   BC_FIELD,   BE_FIELD,
    CA_FIELD,   CC_FIELD)
    VALUES (21, 22, 23, 'B1', 'B3', 'B5','A2', 'C2');
END
```

- The value could be set to the SQL NULL value when a record is updated:

```
SQL
  UPDATE ADANC1 SET AB_FIELD=NULL
  WHERE  AA_FIELD = 21;
END
```

- These fields can be used in search criteria for selecting records that either *have or do not have* SQL NULL values, as in the following examples:

```
SQL
  SELECT AA_FIELD, AB_FIELD, AC_FIELD, AD_FIELD, AE_FIELD FROM ADANC1
  WHERE  AB_FIELD IS NULL AND AA_FIELD GE 10;
END

SQL
  SELECT AA_FIELD, AB_FIELD, AC_FIELD, AD_FIELD, AE_FIELD FROM ADANC1
  WHERE  AB_FIELD IS NOT NULL;
END

TABLE FILE ADANC1R
  PRINT AA_FIELD AB_FIELD AC_FIELD AD_FIELD AE_FIELD G1_GROUP
  WHERE AB_FIELD EQ MISSING;
END
```

- Subdescriptors and superdescriptors can be used only in search criteria. Their values cannot be reported. Here are several examples:

```
SQL
  SELECT AA_FIELD, AB_FIELD, AC_FIELD, AD_FIELD, AE_FIELD FROM ADANC1
  WHERE  S1_FIELD = 'A1A2A4' AND AA_FIELD > 10;
END

SQL
  SELECT AA_FIELD, AB_FIELD, AC_FIELD, AD_FIELD, AE_FIELD FROM ADANC1
  WHERE  S1_FIELD IS NOT NULL AND AA_FIELD > 10;
END
```

- On the mainframe platform, subdescriptors and superdescriptors defined with the NC=YES and MISSING=ON parameters cannot be used to search for an SQL NULL value, as that causes an Adabas RC 61.

Reference: **Master File Field Attributes**

The following field attributes describe Adabas data segments.

Field Attribute	Description
FILE	Master File name. It may or may not match the file name in the Adabas DBMS.
SUFFIX	Always ADBSINX.
SEGNAME	Segment names in the description generated by the Create Synonym facility. They follow a logical format to provide uniqueness within the file.
FIELD	Field name from the Predict dictionary (if used) or generated automatically (xx_FIELD).
GROUP	Identifies fields described as simple groups or PE groups. Is the field name from the Predict dictionary (if used) or generated automatically (xx_GROUP).
ALIAS	Actual Adabas short name. It can be detailed for counter fields without standard length. For order fields, it has the value ORDER.
USAGE	USAGE format and length of the field. This attribute determines how the value is displayed in reports. Values are determined based on ACTUAL format and length and then may be detailed by value from the Predict dictionary (if used). For fields without standard length, it has the maximum value for the format.
ACTUAL	Field standard format and length as stored in the Adabas Field Definition Table (FDT) for a given file. For fields without standard length, it has the maximum value for the format. For fields with format A and option LA (Long Alphanumeric, used only for z/OS) the ACTUAL length must have a value greater than 253. Value 500 is assumed.

Field Attribute	Description
INDEX=I	Indicates the field is a superdescriptor, subdescriptor, or simple descriptor.
TITLE	Column heading used in reports. This attribute is included only if the Predict dictionary is used and the field has a column heading value in the Predict dictionary.
GROUP	Identifies fields described as simple groups or PE groups.
$GRMU	Group that contains a PE group or an MU (multi-value) field.
$2LONG	Group field whose total length exceeds the maximum of 4096 characters supported by the server.
$PEMU	PE group that contains an MU (multi-value) field or is a group field that contains a PE group or an MU field. In both cases, the field cannot be used to retrieve data using the adapter.
$FIELD	Field that belongs to a commented group or superdescriptor composed of partial fields (NOP).

Reference: **Access File Attributes**

The following Access File attributes describe Adabas data segments:

Attribute	Description
RELEASE	Adabas release. If RELEASE is 5 or less, then FILENO can be 1–255 and DBNO can be 0–255. If RELEASE is 6 or greater, then the adapter supports two-byte FILENO (1–5000) and DBNO (1–65535).
OPEN	Determines whether the adapter should issue Adabas OPEN and CLOSE calls for each report request.
SEGNAM	Segment name as described in the Master File.

Attribute	Description
ACCESS	Access method for the segment. ADBS indicates segments that contain non-repeating data. PE indicates segments that describe periodic groups. MU indicates segments that describe multi-value fields.
DBNO	Adabas database number. On UNIX and Windows, and OpenVMS, this attribute must be included in the Access File. DBNO depends on RELEASE. If RELEASE is 5 or less, DBNO can be 0–255. If RELEASE is 6 or greater, DBNO can be 1–65535.
FILENO	Adabas file number. If RELEASE is 5 or less, FILENO can be 1–255. If RELEASE is 6 or greater, FILENO can be 1–5000.
CONNECTION	Is the parameterized form of the logical Connection Name defined in the adapter connection attributes. It is usually created as &&ADA_CONNECTION. It can be re-assigned in an a focexec through a -SET command. For example: `-SET &&ADA_CONNECTION=ADA814`
UNQKEYNAME	Field name that will be used as a unique key during file updating. The first unique indexed field from the root segment is used. If that does not exist, then a value from SEQFIELD is used.
CALLTYPE	FIND. Creates an additional commented line with CALLTYPE=RL if the Predict file view contains SEQFIELD data or the root segment contains an indexed field. The line with the preferable value of CALLTYPE can be chosen depending on the method used to retrieve the data from the database.
SEQFIELD	SEQFIELD name taken from the Predict file view, if used, or the first unique indexed field, or the first indexed field from Root segment. If CALLTYPE=FIND, then no value is entered for SEQFIELD in the Access File.
FIELD	Describes a descriptor, superdescriptor, subdescriptor, phonetic descriptor, hyperdescriptor, or a component of a superdescriptor.

Attribute	Description
TYPE	Type of descriptor for FIELD. Values are SPR for superdescriptors, NOP for subdescriptors or superdescriptors composed of partial fields, PDS for phonetic descriptors, HDS for hyperdescriptors, or DSC for a simple descriptor.
NU	Indicates if the field uses null suppression. Values are YES or NO.
NC	Specifies SQL null representation (NC stands for "not counted"). When on, forces the Create Synonym facility to insert "MISSING=ON" for this field in the associated MFD. NN option below further qualifies the SQL interpretation.
NN	NN "not null" or "null value not allowed" option indicates that an associated NC field must have a value (including zero or blank) defined; it cannot contain "no value.
PASS	Adabas password for the file. This attribute can be added manually.
KEYFLD	Only for child segments. This is the common field in the parent segment, which is used to relate the two files.
IXFLD	Only for child segments. This is the common field in the child segment, which is used to relate the two files.

Reference: **Comments in Master and Access Files**

The generated Master File and Access File contain several commented entries. They are provided for the client and are not used by the server. Comments are provided for the:

- CREATE SYNONYM time stamp and creator's User ID (included in both the Master and Access Files).
- Predict file name, file number, and DBID, if Predict is used.
- Headers before special fields: super/sub/hyper/phonetic descriptors or fields without standard length.
- Fields not supported by the adapter.

- Component fields of superdescriptors that are composed of partial fields (described as NOP type).

Reference: **Usage of Master File Fields With ISN Support**

The Adapter for Adabas can employ a data retrieval strategy through Read Logical by ISN (L1) calls. It can be used to determine the Internal Sequence Number (ISN) of a record that was read or that will be inserted into an Adabas file.

ISN-based access is applicable only if an ISN field is described in the Master File. This field has a field name that is user-defined and an ALIAS of ISN. The Usage format is I and the Actual format must be I4. This field can be defined only in segments that contain non-repeating data (that is, using an access method defined in the Access File as ADBS):

```
FIELD=ISN_FIELD, ALIAS=ISN, I10, I4, $
```

Equality tests on the ISN field can be used to retrieve a single record when the:

- Report request contains an equality operator in the selection test on an ISN list.

  or

- ISN field is used as the cross-referenced field in the Join to an Adabas file.

Adabas returns the Response Code 113 if the record with the ISN defined in the test is not present in the Address Converter for the file. The adapter returns the following message: "Record is not found".

The Adapter for Adabas uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when the:

- Report request does not contain optimizable selection criteria on an inverted list.
- Access File contains a SEQFIELD value for the segment whose value is equal to the ISN field name.

For example, the Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=ISN_FIELD ,$
```

Adabas returns each record in ascending order by the value of the ISN.

Example: **Equality Test on ISN Field**

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST
WHERE ISN_FIELD = 1100;
```

Note: Multifetch option will be suppressed for this call.

Read Logical by ISN (L1) calls to retrieve a subset of records for the entry segment when the:

- Report request contains the GE/GT relational operator in the selection test on an ISN list.
- Report request does not contain any selection tests on an inverted list.
- Access File contains a SEQFIELD value for that segment.

Example: Inequality Test on ISN Field

Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS, ... ,SEQFIELD=AA_FIELD ,$
```

Read Logical by ISN (L1) is used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100;
```

Reference: ISN for Insert

After issuing an Insert request on a file with an ISN field in the Master File, the resulting ISN generated by Adabas is displayed by the message FOC4592:

```
(FOC4592) RECORD IS INSERTED WITH ISN : nnnnn
```

Example: Issuing an Insert Request

```
SQL
 INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD)
VALUES ('11111111', 'TAMPA');
END


ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 11
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT


TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

You can assign a value for the ISN field if the Insert request contains the ISN field and the assigned value is not 0.

The adapter will issue an Adabas N2 Direct Call to assign this ISN value to the inserted record. Adabas returns Response Code 113 if this value was already assigned to another record in the file or if it is larger than the MAXISN in effect for the file.

Reference: **Master File Fields and GFBID Support**

The Adapter for Adabas can optimize the performance of queries that use the same Adabas field lists repeatedly. Field lists are generated in Adabas format buffers, and can be retained.

Global Format Buffer ID (GFBID) support is applicable only if a GFBID field is present in the Master File. This field has a user-defined field name, and an ALIAS of GFBID. This field is used to determine the Global Format Buffer ID that will be defined in read requests with identical field lists for the same database. The GFBID field has a USAGE format of A8 and an ACTUAL format of A8. This field can be defined only in segments that contain non-repeating data (that is, the access method defined in the Access File is ADBS).

Example: **Using GFBID Syntax in a Master File**

```
FIELD=GID_FIELD, ALIAS=GFBID, A8, A8, $
```

Note:

- If the field list is changed but the same GFBID is used in a request, then incorrect results may be displayed. In some cases, a message about the possible mismatch between the Field Definition Table (FDT) and Master File will be issued.
- If two requests have the same list of selected fields but different fields are used in selection criteria, then the requests must use different GFBID values.

The GFBID field can be defined in the Master File manually or using the Create Synonym facility with option PARMS GFBID.

Reference: **Using the GFBID Parameter With the Create Synonym Facility**

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

The value of GFBID can have up to 8 bytes and must start with a digit or uppercase character.

If GFBID is used in a request against a file that contains simple PE/MU segments, it will be applied to Adabas calls to get the field values from these segments. The GFBID value will be adjusted for each non-ADBS segment, that is, a segment number will be placed in the last (8th) byte of the GFBID field.

Calls to a PE (periodic element) with MU (multi-value) child or an MU with PE parent segment will not use GFBID values.

Note: To make the GFBID value unique within Adabas, do not use 8 byte GFBID values for these types of calls.

- If a GFBID field is defined in the Master File, it can be used in a request as part of the selection test. The adapter will take the GFBID value from the selection criteria, deactivate the field, and remove it from the match array. The GFBID value is placed in the ADD5 field of the Adabas Control Block for the request.
- If the GFBID is applied to an Adabas call, then field name ADD5 appears in the adapter trace. GFBID values that were issued in all requests to the database are accumulated in the Adabas internal queue. These values can be removed from the queue by issuing the following adapter SET command:

  ```
  ENGINE ADBSINX SET GFBID_OFF ALL/<value> DBID <number>
  ```

When a single <value> is in the SET command, all values issued for child segments are removed as well. If <value> contains blanks, then value must be enclosed in single quotation marks.

ALL clears all GFBID values.

Example: **Using GFBID Syntax**

Master File ADATEST contains:

```
FIELD=GID_FIELD, ALIAS=GFBID, A8, A8, $
```

Request 1. Global ID = ABC1:

```
SQL
SELECT AA_FIELD, AJ_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD,
AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
PAGE 1

 AA_FIELD AJ_FIELD     ISN_FIELD AQ0201_OCC AR_FIELD AS_FIELD AT_FIELD
 -------- --------     --------- ---------- -------- -------- --------
 30034517 DERBY        1105      1          UKL      6500     750
 30034517 DERBY        1105      2          UKL      6800     1240
 30034517 DERBY        1105      3          UKL      7100     1345
 30034517 DERBY        1105      4          UKL      7450     1450
 30034517 DERBY        1105      5          UKL      7800     1700
 50005600 VIENNE       1106      1          FRA      165810   10000
 50005300 PARIS        1107      1          FRA      166900   5400

 NUMBER OF RECORDS IN TABLE= 7 LINES= 7
```

Request 2. Global ID = ABC1, but field's list is changed:

```
SQL
SELECT AA_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD, AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
(FOC4567) POSSIBLE MISMATCH BETWEEN FDT AND MFD : file=19879548
(FOC4591) ERROR RETURN ON READ BY ISN : S01 /999:ffffffcd001d00

 NUMBER OF RECORDS IN TABLE= 0 LINES= 0
```

Request 3. Remove value 'ABC1' from Adabas:

```
ENGINE ADBSINX SET GFBID_OFF ABC1 DBID 3
```

Request 4. Repeat request 2.

```
SQL
SELECT AA_FIELD, ISN_FIELD, AQ0201_OCC, AR_FIELD, AS_FIELD, AT_FIELD
FROM EMPLOYEES
WHERE ISN_FIELD > 1100 AND GID_FIELD = 'ABC1';
END
```

Result:

```
PAGE 1

 AA_FIELD ISN_FIELD AQ0201_OCC AR_FIELD AS_FIELD AT_FIELD
 -------- --------- ---------- -------- -------- --------
 30034517 1105      1          UKL      6500     750
 30034517 1105      2          UKL      6800     1240
 30034517 1105      3          UKL      7100     1345
 30034517 1105      4          UKL      7450     1450
 30034517 1105      5          UKL      7800     1700
 50005600 1106      1          FRA      165810   10000
 50005300 1107      1          FRA      166900   5400

 NUMBER OF RECORDS IN TABLE= 7 LINES= 7
```

Request 5. Remove all GFBID values for Database 3 from Adabas:

```
ENGINE ADBSINX SET GFBID_OFF ALL DBID 3
```

Overview of Master and Access Files

The server processes a report request with the following steps:

1. It locates the Master File for the Adabas file. The file name in the report request is the same as the z/OS PDS member name.
2. It detects the SUFFIX attribute, ADBSINX, in the Master File. Since this value indicates that the data resides in an Adabas file, it passes control to the adapter.
3. The adapter locates the corresponding Access File and uses the information contained in both the Master and Access Files to generate the Adabas direct calls required by the report request. It passes these direct calls to the Adabas DBMS.
4. The adapter retrieves the data generated by the Adabas DBMS and returns control to the server. For some operations, the server may perform additional processing on the returned data.
5. Beginning with Version 7 Release 7.4 of the server, the same adapter supports access to the database with different SVCs through different connections (see Connection Name in *Connection Attributes for Adabas* on page 132).

   The CREATE SYNONYM facility will place the FILENO, DBNO, and CONNECTION Access File attributes in parameterized form with default values that were provided when the synonym was created.

   These values can be changed later with a FOCEXEC by using a -SET command. For example:

   ```
   -SET &&ADA_CONNECTION=ADA814;
   -SET &&ADA_DBNO=123;
   -SET &&ADA_FILNO=1234;
   ```

Access and Master Files created in previous versions can be used without any changes, except in the case where legacy files are going to be used to access Adabas databases with different SVCs . In those cases, the new CONNECTION attribute has to be placed into the old Access File, as shown in example above and the value of &&ADA_CONNECTION must correspond to the Connection Name from the connection string in server profile.

If CONNECTION is not found among the connection strings, the adapter will issue error message FOC4515.

Master Files for Adabas

Standard Master File attributes are generally used to describe Adabas files, but there are concepts in Adabas file processing that require special terminology at the field and segment level. On z/OS, the Master File is a member of a PDS allocated to DDNAME MASTER.

The server permits a great deal of flexibility in its Master Files. For the Adapter for Adabas, you need to describe only the Adabas fields actually used in reporting applications, omitting any unnecessary Adabas fields. (Note the exception that if a PE group is included in the Master File, all fields of the PE group must be included. See *Field Attributes in Master Files* on page 170 and *Describing Multi-Value Fields Within Periodic Groups* on page 197 for additional information.) Additionally, describing the same Adabas file in multiple ways enables you to access that same file in its different configurations within one procedure.

Master Files have three parts: file declaration, segment declaration, and field declaration, each of which is explained in these topics. Concepts that are specific to Adabas files, for which the server requires unique syntax in the Master File, are also explained.

The following is an example of the Master File for VEHICLES.

```
$$$ CREATED ON 12/10/03 AT 10.17.27 BY PMSMJB
FILENAME=ADACAR,SUFFIX=ADBSINX,$

$ ADABAS FILE = VEHICLES-FILE                      DICTIONARY = 6
SEGNAME=S01     ,SEGTYPE=S,$
 FIELD=REG_NUM                    ,ALIAS=AA     ,A15   ,A15  ,INDEX=I,$
 FIELD=CHASSIS_NUM                ,ALIAS=AB     ,I9    ,I4   ,$
 FIELD=PERSONNEL_ID               ,ALIAS=AC     ,A8    ,A8   ,INDEX=I,$
 GROUP=CAR_DETAILS                ,ALIAS=CD     ,A50   ,A50  ,$
  FIELD=MAKE                      ,ALIAS=AD     ,A20   ,A20  ,INDEX=I,$
  FIELD=MODEL                     ,ALIAS=AE     ,A20   ,A20  ,$
  FIELD=COLOR                     ,ALIAS=AF     ,A10   ,A10  ,INDEX=I,$
 FIELD=YEAR                       ,ALIAS=AG     ,P2    ,Z2   ,$
 FIELD=CLASS                      ,ALIAS=AH     ,A1    ,A1   ,INDEX=I,$
 FIELD=LEASE_PUR                  ,ALIAS=AI     ,A1    ,A1   ,$
 FIELD=DATE_ACQ                   ,ALIAS=AJ     ,P6    ,Z6   ,$
$GRMU=CAR_MAINTENANCE             ,ALIAS=AK     ,A11   ,A7   ,$
  FIELD=CURR_CODE                 ,ALIAS=AL     ,A3    ,A3   ,$
  FIELD=MAINT_COST_CNT            ,ALIAS=AMC    ,I4    ,I2   ,$
 FIELD=DAT_ACQ_DESC               ,ALIAS=AN     ,A4    ,A4   ,INDEX=I,$
 GROUP=MODEL_YEAR_MAKE            ,ALIAS=AO     ,A28   ,A22  ,INDEX=I,$
  FIELD=YEAR_S02                  ,ALIAS=AG     ,P2    ,Z2   ,$
  FIELD=MAKE_S02                  ,ALIAS=AD     ,A20   ,A20  ,INDEX=I,$

SEGNAME=AM0101  ,SEGTYPE=S,PARENT=S01  ,OCCURS=AMC,$  MAX= 60
 FIELD=MAINT_COST                 ,ALIAS=AM     ,P7    ,P4   ,$
 FIELD=AM0101_OCC                 ,ALIAS=ORDER  ,I4    ,I2   ,$
```

File Attributes in Master Files

Master Files begin with a file declaration that names the file and describes the type of data source. The file declaration has two attributes, FILENAME and SUFFIX.

The syntax is

```
FILE[NAME]=filename, SUFFIX=ADBSINX [,$]
```

where:

filename

Is a one- to eight-character name that complies with server file naming conventions.

ADBSINX

Is the value that specifies the Adapter for Adabas.

FILENAME

The FILENAME (or FILE) is any valid name from one to eight characters long.

In the z/OS environment, it is a member of a partitioned data set allocated to the DDNAME MASTER.

SUFFIX

The value for the SUFFIX attribute is always ADBSINX, which is the name of the Adapter for Adabas program that the server loads to read an Adabas database.

Example: **File Declaration**

The following example is a typical file declaration:

```
FILENAME=ADACAR, SUFFIX=ADBSINX,$
```

Segment Attributes in Master Files

Each segment declaration describes one Adabas record or a subset of an Adabas record (called a logical record type). Each logical record type described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

The syntax is

```
SEGNAME=segname, SEGTYPE=segtype [,PARENT=parent]
                                 [,OCCURS=occursname][,$]
```

where:

segname

Is a unique one- to eight-character name that identifies the segment.

segtype

Identifies the characteristics of the segment. Possible values are:

S which indicates multiple instances of a descendant segment are related to a given parent.

SO which indicates the Write Adapter for Adabas (for the Read Adapter for Adabas, SEGTYPE=S is sufficient).

U which indicates a single instance of data for a descendant segment is related to its parent.

KL which indicates multiple instances of a cross-referenced descendant segment are related to a given parent.

KLU which indicates a single instance of a cross-referenced descendant segment is related to its parent.

parent

Is the name of the parent segment in the Master File.

occursname

Indicates a data field in the parent segment that contains the number of occurrences of the descendant segment. Its value is derived from the two-character internal Adabas name (ALIAS field on the server) appended with the letter C.

SEGNAME

The SEGNAME (or SEGMENT) attribute is the name of a group of data fields that have a one-to-one relationship to each other in the Adabas file.

You may describe an Adabas file in the server hierarchical design to take advantage of multi-value fields and periodic groups. You may have multiple Adabas records described in the same Master File view. Each record is described as a segment with a unique SEGNAME.

Syntax: How to Name a Root Segment in the Master File

For the root segment of the Adabas file, the segment name generated is

S*nn*

where:

nn

Is a two-digit number indicating the order in which the file was selected.

The first file (selected as the root from the File Selection Menu) has SEGNAME=S01. Subsequent files used in the same description (selected as children from the File Selection Menu) have SEGNAME=S02, SEGNAME=S03, and so on.

Syntax: How to Name a Segment for a PE Group and MU Field

The segment names for PE (periodic element) groups and MU (multi-value) fields have the format

aammnn

where:

aa

Is the Adabas field name.

mm

Is a two-digit number that indicates the order in which the PE group or MU field appears in the PREDICT description of the file.

nn

Is the order number used for the root segment.

Example: **Naming a Segment for a PE Group and MU Field**

Using the syntax aammnn, SEGNAME=BE0201 describes the segment for the field BE, which is the second (02) PE group or MU field described in the first segment (01).

SEGTYPE

The SEGTYPE attribute identifies the basic characteristics of related or cross-referenced segments. Cross-referencing is a method of accessing information from two or more different files or segments to use in a single report request.

SEGTYPE

Is the root segment and cross-reference segment type. The SEGTYPE is S0 for the root file and children with a non-unique IXFLD. The SEGTYPE is U for a child with a unique IXFLD. The SEGTYPE for all PE and MU fields is S0.

PARENT

Any segment except the root is a descendant, or child, of another segment. The PARENT attribute supplies the name of the segment which is the logical parent or owner of the current segment. If no PARENT attribute appears, the default is the immediately preceding segment. However, it is highly recommended to include the parent. The PARENT name is the one- to eight-character SEGNAME of a previous segment.

PARENT

Is the value of SEGNAME of the parent record.

OCCURS

For more information on the segment attribute OCCURS, which applies to repeating fields and groups, see *Describing Multi-Value Fields and Periodic Groups With the OCCURS Attribute* on page 195.

OCCURS

Indicates the number of occurrences of a PE group or MU field. This attribute contains the Adabas field name of the PE group or MU field with the suffix C, which is the ALIAS of the counter field in the parent segment.

Example: Segment Declaration

The following two examples are typical segment declarations:

```
SEGNAME=S01         ,SEGTYPE=S      ,$
SEGNAME=AM0101      ,SEGTYPE=S      ,PARENT=S01     ,OCCURS=AMC    ,$
```

Field Attributes in Master Files

Field declarations describe the fields in each file. For the simplest file structures (single-segment, fixed-length files), you need to describe only the Adabas fields you actually use for reporting purposes.

Put the fields in any order within their segment, except in the case of fields within periodic groups (PE). The PE group must have all fields defined and in the proper order.

The syntax is

```
FIELD[NAME]=fieldname, ALIAS=alias, [USAGE=]display, [ACTUAL=]format,$
```

where:

fieldname

Is the 1- to 66-character name that is unique to the Master File.

alias

Is the two-character internal Adabas field name.

display

Is the server display format for the field.

format

Is the server definition of the Adabas field format and length.

Note: Field declarations are always terminated with ,$.

There are additional attributes that apply to superdescriptor, cross-referenced, repeating, and other types of fields.

FIELDNAME

The field name appears as a column heading when you include the field in a report request. You can change the default by using AS or NOPRINT in the report or by using TITLE in the Master File.

Field names are unique names of up to 66 characters. The Master File field name does not need to be the same as the Adabas field name. Field names can include any alphanumeric characters, but the first character must be a letter from A to Z. Avoid embedded blanks and special characters.

Since all references to data on the server are through field names or aliases, the field names should be unique within a Master File. This requirement is true even when the same Adabas record is included as two differently structured segments in the same Master File. You must specify different field names in the two segments or the server uses the first one in the Master File unless you qualify which field you want, for example, file.field, segment.field.

ALIAS

The ALIAS for a single Adabas field *must* contain the two-character internal Adabas field name. The adapter uses it to generate direct calls to the Adabas DBMS. The format rules are as follows:

- The ALIAS equals the two-character internal Adabas field name, for example, AM.
- The letter C is appended to the two-character internal Adabas field name if the field is the counter field for a PE group or MU field, for example, AMC.
- The ALIAS is the two-character internal Adabas field name appended with the digit 1 if an application calls for only the first occurrence of a repeating field, for example, AM1.
- The ALIAS can be used to reformat the length of the Adabas field. See *Using the ALIAS to Reformat Adabas Fields* on page 171 for examples using the reformatting option.

Note: Because of the requirement that the ALIAS in a Master File be the same as the two-character internal Adabas field name, your Master File may include duplicate ALIAS values. In that case, the field name is used to differentiate between the segments.

Using the ALIAS to Reformat Adabas Fields

The ALIAS field allows reformatting of the field for a different view. Several Adabas data formats have lengths that exceed the supported maximum length of data formats on the server. You can use the ALIAS to convert this data to an acceptable ACTUAL format and length.

When a field is specified in a retrieval request, the value in the ALIAS is passed to Adabas in the Format buffer. Adabas uses the ALIAS to process the field and return the data to the server.

To use the reformatting option, define the ALIAS using length and format values that indicate how you want to reformat the field.

The syntax is

```
ALIAS='xx,l,f'
```

where:

xx

Is the two-character internal Adabas field name.

l

Is the new length of the field value in number of bytes.

f

Is the new Adabas format of the field value.

The entire value must be enclosed in single quotation marks.

For example, if you want to use only the first three characters in a six-byte field, use the following syntax in your Master File:

```
FIELD=L_DBNR, ALIAS='AU,3,P', USAGE=P3, ACTUAL=P3  ,$
```

In this case, the Adapter for Adabas will read only the first three bytes (L_DBNR is a six-byte field with the Adabas description: 01, AU, 6, U).

In the next example, to convert an Adabas binary field format of ten bytes (Adabas description: 01, BB, 10, B) to an alphanumeric format large enough to accommodate the Adabas value, you could use the following syntax in your Master File:

```
FIELD=TESTFLD, ALIAS='BB,10,A', USAGE=A20, ACTUAL=A10  ,$
```

Notice that you must also change the USAGE and ACTUAL values to reflect the changes in the format. Refer to the chart in *ACTUAL* on page 173 for the acceptable USAGE and ACTUAL values along with the corresponding Adabas codes.

Note: Groups cannot be reformatted. This formatting is valid only for elementary fields.

USAGE

The USAGE attribute enables you to describe how you want fields displayed on the screen, printed in reports, or used in calculations. This attribute includes the data field type, length, and any applicable display options when the field values are printed. FORMAT is a synonym for USAGE.

The syntax is

```
USAGE=tllleeeee
```

where:

t

Is the field type.

lll

Is the number of positions needed to display the field.

eeeee

Are the display options.

The value that you specify for field length is the number of alphanumeric characters or numeric digits that the server allocates for the field in any display or report. The specified value excludes the editing characters, such as comma, $, and so on. Display options control how a field value is printed.

ACTUAL

The ACTUAL attribute describes the length and type of the Adabas field in storage. It is used to describe the format of fields from external data files only, and must be included in the Master Files needed for the Adapter for Adabas. The source of this information is your existing Adabas Field Definition Table (FDT).

The syntax is

ACTUAL=*tlll*

where:

t

Is the server field type of the Adabas field.

lll

Is the storage length of the Adabas field.

The server permits the following conversions from ACTUAL format to USAGE (display) format:

Adabas Format Type	Server ACTUAL Type	Server ACTUAL Length*	Server USAGE Type	Server USAGE Length*	Description
A	A	1-256	A	1-256	Alphanumeric

Adabas Format Type	Server ACTUAL Type	Server ACTUAL Length*	Server USAGE Type	Server USAGE Length*	Description
G	F D	4 8	F D	1-9 1-15	Float Single-Precision Float Double-Precision
F	I	2, 4	I	1-9	Integer
B 1-4 5-6 7-126	 I P A	 1-4 7-8 9-126	 I P A	 1-9 12-15 14-252	 Integer Values Packed Decimal Alpha
P	P	1-15	P	1-33.*n*	Packed Decimal
U	Z	1-29	A or P	1-33.*n*	Zoned

* Lengths are measured as follows:

- ACTUAL is in number of bytes.
- USAGE allows space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

The following table shows the codes for the types of data the server reads:

ACTUAL Type	Description
A*n*	Alphanumeric characters A to Z, 0 to 9, and the special characters in the EBCDIC display mode, where *n* = 1 to 256 bytes.
D8	Double-precision, floating-point numbers, stored internally in eight bytes.
F4	Single-precision, floating-point numbers, stored internally in four bytes.
I*n*	Binary integers, where *n* = 1 to 4.
P*n*	Packed decimal data, where *n* = 1 to 15 bytes.

ACTUAL Type	Description
Z*n*	Zoned decimal data, where *n* = 1 to 30 bytes. Represent the field as Z*m.n*, where *m* is the total number of digits, and *n* is the number of decimal places (for example, Z6.1 means a six-digit number with one decimal place).

INDEX

For more information on the field attribute INDEX, which applies to descriptors, see *Specifying INDEX=I* on page 184.

GROUP

For more information on the field attribute GROUP, which applies to superdescriptors, see *Specifying Superdescriptors Using the GROUP Attribute* on page 185.

Example: **Field Declaration**

The following is a typical field declaration:

```
FIELD=MODEL ,ALIAS=AE ,USAGE=A20 ,ACTUAL=A20 ,$
```

Access Files for Adabas

An Access File describes the physical attributes of the Adabas file, such as the database number, file number, descriptors, and security. See *Overview of Master and Access Files* on page 164 for more information about Master and Access Files.

Access Files store database access information used to translate report requests into the required Adabas direct calls.

In the z/OS environment, the Access File is located in a PDS allocated to the DDNAME ACCESS. Each member is a separate Access File.

An Access File consists of 80-character records in comma-delimited format. A record in an Access File contains a list of attributes and values, separated by commas and terminated with a comma and dollar sign (,$). This list is free-form and spans several lines if necessary. You can specify attributes in any order.

The server reads blank lines, and lines starting with a dollar sign in column one, as comments.

Every Access File contains a release declaration and at least one segment declaration. Release declarations and segment declarations each have their own set of attributes. These attributes are discussed in the following topics.

The following is a sample Access File:

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                          DICTIONARY =
SEGNAM=S01   ,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
              SEQFIELD=PERSONNEL_ID,$
FIELD=DEPT_PERSON                          ,TYPE=SPR,$
FIELD=DEPT                                 ,TYPE=DSC,NU=YES,$
FIELD=NAME                                 ,TYPE=   ,NU=NO,$
 FIELD=LEAVE_LEFT                          ,TYPE=SPR,$
 FIELD=LEAVE_DUE                           ,TYPE=   ,NU=YES,$
 FIELD=LEAVE_TAKEN                         ,TYPE=   ,NU=YES,$
FIELD=DEPARTMENT                           ,TYPE=NOP,NU=NO,$
SEGNAM=AI0101,ACCESS=MU  ,FILENO=001,$ ADDRESS_LINE
SEGNAM=AQ0201,ACCESS=PE  ,FILENO=001,DBNO=1,$ INCOME
SEGNAM=AT0301,ACCESS=MU  ,FILENO=001,DBNO=1,$ BONUS
SEGNAM=AW0401,ACCESS=PE  ,FILENO=001,$ LEAVE_BOOKED
SEGNAM=AZ0501,ACCESS=MU  ,FILENO=001,$ LANG
```

Release Declaration

The release declaration must be the first uncommented line of the Access File. It identifies the release of Adabas and indicates whether the Adapter for Adabas issues Adabas OPEN and CLOSE calls for each report request.

The syntax is

```
RELEASE=relnum [,OPEN={YES|NO}] ,$
```

where:

relnum

Is the Adabas release number.

OPEN

Specifies whether the adapter issues Adabas OPEN and CLOSE calls to Adabas in each report request. Possible values are:

YES which indicates that Adabas OPEN and CLOSE calls are issued. YES is the default value.

NO which indicates that Adabas OPEN and CLOSE calls are not issued.

RELEASE

The first declaration in the Access File contains the Adabas release number with the attribute RELEASE. Specify the full release number. The value is for documentation only.

The following example illustrates the use of the RELEASE attribute to specify the Adabas release number:

```
RELEASE=6 ,$
```

OPEN

The OPEN attribute specifies whether the Adapter for Adabas issues Adabas OPEN and CLOSE calls to Adabas in each report request. Specifying YES or NO achieves the following results:

Value	Result
YES	The adapter issues an OPEN call to Adabas at the start of a call set initiated by a retrieval request. An Adabas CLOSE is issued at the end of retrieval for the request. YES is the default value. `RELEASE=6 , OPEN=YES ,$`
NO	Adabas OPEN and CLOSE calls are not issued for any retrieval request. `RELEASE=4.1, OPEN=NO ,$`

Accept the default value (YES) unless you are using an Adabas release lower than 5.0.

Note: To reduce overhead incurred in opening and closing databases when processing requests from multiple agents, the Adapter for Adabas allows opened databases to remain open until one of the following actions occurs:

- A request changes the database, the file, the database mode (close) or its access mode (read/write).
- The last agent disconnects.
- The last open database is closed explicitly.

Segment Attributes in Access Files

Segment declarations contain detailed information about each segment of an Adabas file. The segment declaration attributes specify the segment name, the file and database numbers, and the Adabas password, as well as retrieval options.

The syntax is

```
SEGNAM=segname, ACCESS=access, FILENO=file_number [,DBNO=database_number]
[,CALLTYPE={FIND|RL}] [,USE={FIND|ANY}] [,SEQFIELD=seqfield] [,PASS=password]
[,FETCH={ON|OFF}] [,FETCHSIZE={n|MAX}] [,UNQKEYNAME=unique_field] [,WRITE=YES|NO],$
```

where:

segname

Is the SEGNAME value from the Master File.

access

Specifies the segment type that the adapter uses for each segment. Possible values are:

ADBS which is for a segment that contains only non-repeating fields. ADBS is the default value.

PE which is for a segment that is a periodic group in the same file as the parent segment.

MU which is for a segment that is a multi-value field in the same file as the parent segment.

file_number

Is the number that identifies an Adabas file.

database_number

Is the number that identifies an Adabas database.

CALLTYPE

Indicates the type of data retrieval call constructed by the adapter. Possible values are:

FIND in which a FIND call is issued when there are one or more WHERE or IF statements in a retrieval request detected against fields defined with INDEX=I. FIND is the default value.

RL in which a Read Logical (RL) call is issued when there are one or more WHERE or IF statements against fields defined with INDEX=I.

Note: See your database administrator for the most efficient CALLTYPE to be used at your site.

USE

Indicates a method for handling unreadable sub or superdescriptor fields that are derived from PE or MU access types and defined in the root segment. Possible values are:

FIND indicates that the S1 command will be issued against such a field even if CALLTYPE is RL.

ANY indicates that fields are not of the sub/superdescriptor types. ANY, or the absence of a USE= entry, are the default values.

seqfield

Provides a default index which controls Read Logical (RL) retrieval when there is no IF or WHERE selection test.

password

Is the Adabas password for the file.

FETCH

Indicates whether to use the Fetch feature. Possible values are:

ON which sets the Fetch feature on for the user session. ON is the default value.

OFF which sets the Fetch feature off for the user session.

If you include this attribute, you must also include FETCHSIZE (see below).

FETCHSIZE

Sets the number of records per buffer. Possible values are:

n which is the number of records per buffer (1 to 32K). 10 is the default value.

MAX which is automatically calculated by the adapter to allow the maximum number of records that fit in a 32K buffer.

UNQKEYNAME

Is the attribute that indicates the unique key. It does not necessarily define the key described in the ADABAS FDT with the UQ option.

WRITE

Is the attribute that indicates the WRITE capability. NO is the default value.

Note: FETCH and FETCHSIZE are applicable only to segments described as ACCESS=ADBS. FETCH and FETCHSIZE can be set dynamically to override the Access File settings.

Special attributes that apply to embedded JOINs are explained in *Implementing Embedded JOINs: KEYFLD and IXFLD* on page 190. Attributes that apply to superdescriptors and subdescriptors are explained in *Customizing the Adabas Environment* on page 201.

SEGNAM

The SEGNAM attribute specifies the name of the segment being described. It is the same name as the SEGMENT value in the Master File.

ACCESS

The ACCESS attribute specifies the segment type that the Adapter for Adabas uses for each segment. The values are:

- ADBS, which specifies a segment that contains only non-repeating fields. ADBS is the default value.
- PE, which specifies a segment that is a periodic group in the same file as the parent segment.
- MU, which specifies a segment that is a multi-value field in the same file as the parent segment.

All three access values use the attributes FILENO and DBNO, as shown in this partial Access File for EMPLOYEES with the specification of PE and MU fields.

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                          DICTIONARY =
SEGNAM=S01    ,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
               SEQFIELD=PERSONNEL_ID,$
SEGNAM=AQ0201,ACCESS=PE   ,FILENO=001,DBNO=1,$ INCOME
SEGNAM=AT0301,ACCESS=MU   ,FILENO=001,DBNO=1,$ BONUS
SEGNAM=AW0401,ACCESS=PE   ,FILENO=001,DBNO=1,$ LEAVE_BOOKED
```

ACCESS=ADBS

ACCESS=ADBS specifies the main segment of the file. This segment contains only non-repeating fields.

FILENO

The FILENO attribute specifies the Adabas file number. You need this file number to identify the Adabas file(s) you wish to access. The valid values are 1 to 255 (5000 is the maximum value for a mainframe platform). The FILENO attribute has to be defined for segments with ACCESS=ADBS. If the FILENO attribute is omitted for segments with ACCESS=MU/PE, then it will be taken from the corresponding parent segment.

Suppose you are accessing two segments: the first is in the EMPLOYEE file, and the second is in the INSURANCE file. To describe this situation in the Access File, use the SEGNAM and FILENO attributes as illustrated in *DBNO* on page 181.

DBNO

The DBNO attribute specifies the Adabas database number. It is used when you define files from multiple databases in one Master File. If it is not provided, the DBNO will be read from the DDCARD (on a mainframe platform only). If the DDCARD is not allocated, the adapter uses the default value, DBNO=0.

If the DBNO attribute is omitted for a segment with ACCESS=PE/MU, it will be taken from the corresponding parent segment. The DBNO attribute has to be defined for segments with ACCESS=ADBS on non-mainframe platforms.

The following example illustrates the use of both the FILENO and DBNO attributes to account for the files being in different databases:

```
SEGNAM=ONE   ,FILENO=1  ,DBNO=1 ,... ,$
SEGNAM=PAY   ,FILENO=2  ,DBNO=3 ,... ,$
```

CALLTYPE

The CALLTYPE attribute affects how the Adapter for Adabas processes WHERE and IF statements on descriptors in report requests and how it retrieves cross-referenced file data from a descendant segment. With CALLTYPE, you optionally specify the type of access to use in order to process the inverted lists for any given segment. Ask your Adabas database administrator to advise you when you are selecting a CALLTYPE.

The possible values are:

FIND

An Adabas FIND call is issued when there are one or more WHERE or IF statements against fields defined to the server with INDEX=I. The search is on specific values of the field.

The Adapter for Adabas can generate Adabas FIND (S1) calls even when non-descriptor fields are used as search criteria. This can occur whenever CALLTYPE=FIND is specified in the Access File, and if you include an IF or WHERE test when referencing a non-descriptor field in your TABLE or TABLEF request.

To turn off this feature, see the topic in *Adapter Navigation* on page 227 that discusses the optimization of the FIND call using non-descriptor fields.

RL

An Adabas Read Logical call is issued when there are one or more WHERE or IF statements against fields defined to the server with INDEX=I.

Note: If you have selected a field defined with TYPE=NOP (for example, subdescriptors), a FIND call is issued even if RL has been specified in cases when:

- SET NOPACCESS FIND was performed.
- The server is running in OpenVMS environment.
- The same Adabas field is defined in the descendant PE/MU segment.

Switching from RL to FIND mode is also performed if a field is defined:

- With TYPE=SPR and the server is running in an OpenVMS environment.
- With TYPE=PDS (phonetic descriptor) or TYPE=HDS (hyperdescriptor).
- In a PE segment (is a part of periodic group).

The adapter uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when the:

- Report request does not contain an optimizable selection test on an inverted list or an ISN list.
- The Access File contains a SEQFIELD value for that segment and this value is equal to the ISN field name.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN.

The first IF statement in a request that refers to a field with INDEX=I determines the inverted list used for the Read Logical access for the root of the subtree.

When retrieving descendant segments, the Adapter for Adabas issues a Read Logical call using the descriptor pointed to by IXFLD. For more information on IXFLD, refer to *Implementing Embedded JOINs: KEYFLD and IXFLD* on page 190.

For CALLTYPE=RL, all other selection tests on descriptor fields specified in your report request are applied after the record is returned.

USE=FIND

The Adapter for Adabas cannot read sub/superdescriptors that are derived from PE/MU access types and defined in a root segment. These types of fields may be used only in an Adabas FIND command that produces a "trusted" answer set. However, you can add the description parameter USE=FIND in the Access File for sub and superdescriptor fields that cannot be read by the adapter.

The following processing rules apply for fields that are defined with this description or that are automatically recognized as a fields derived from PE/MU:

- The Adabas command S1 is issued even if CALLTYPE is RL in the Access File.
- If the field is used in a PRINT/WRITE list, the adapter issues the error message:

  ```
  (FOC4582) THIS NOP/SPR FIELD CAN BE USED IN FIND CALLS ONLY.
  ```

- If the field is used in a complex WHERE condition and cannot be fully converted to the generated FIND call, the adapter issues the error message:

  ```
  (FOC4493) THIS NOP/SPR FIELD CANNOT BE USED IN COMPLEX FIND
  ```

The Create Synonym facility always places sub/superdescriptors derived from PE/MU in a root segment with TYPE=NOP in the Access File. Moreover, the sub descriptors have the description USE=UFIND in the Access File.

SEQFIELD

The SEQFIELD attribute specifies the field you want to use as the sequencing value:

```
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                          DICTIONARY =
SEGNAM=S01,ACCESS=ADBS,FILENO=001,DBNO=1,CALLTYPE=RL,
           SEQFIELD=PERSONNEL_ID,$
```

PASS

The PASS attribute specifies the Adabas password for the file. It is required if the file is password protected.

You must specify the password for each password-protected file you use. For example:

```
SEGNAM=ONE    ,FILENO=1    ,DBNO=1    ,PASS=ONEXX ,...,$
SEGNAM=PAY    ,FILENO=2    ,DBNO=3    ,PASS=USER2 ,...,$
```

FETCH

The FETCH attribute indicates whether to use the Fetch feature for segments described as ACCESS=ADBS in the Access File. Valid values are ON (default) or OFF.

FETCHSIZE

The FETCHSIZE attribute sets the number of records per buffer. You can supply a specific number or have the Adapter for Adabas automatically calculate the maximum number of records that fit in a 32K buffer.

The following is an example of an Access File that enables the Fetch feature and sets the number of records per buffer to 15.

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                          DICTIONARY =
SEGNAM=S01,ACCESS=ADBS,FILENO=001,CALLTYPE=RL,
           SEQFIELD=PERSONNEL_ID,FETCH=ON,FETCHSIZE=15,$
```

Mapping Adabas Descriptors

Adabas descriptors, superdescriptors, and subdescriptors must be declared in Master and Access Files in certain ways:

- When creating Master Files for fields that are descriptors, use the INDEX=I attribute. The Access File must also contain information about the descriptors, specifying the TYPE as DSC, SPR, or NOP. Identifying descriptors in the Access File directs the server to inverted lists to speed retrieval of these fields.
- Adabas fields described as superdescriptors must be declared in the Master File using the GROUP attribute. In the Access File, the TYPE must be SPR.

Specifying INDEX=I

Descriptors act as key values associated with a single field in an Adabas record. They enable records to be retrieved more efficiently based on the selection of a value, a set of values, or a range of values.

Additionally, all descriptors, superdescriptors, and subdescriptors are used to establish logical relationships between files. There are two methods for establishing logical relationships:

- By establishing the relationship dynamically at run time with the JOIN command. For more information about using Adabas files that are to be cross-referenced using JOIN, see *Adapter Navigation* on page 227.
- By using a static cross-reference in the Master and Access Files. For information on how to define shared files, see *Implementing Embedded JOINs: KEYFLD and IXFLD* on page 190.

Adabas descriptor fields are identified in the Master File using the INDEX=I attribute. For example:

```
FIELD=MAKE ,ALIAS=AD ,USAGE=A20 ,ACTUAL=A20 ,INDEX=I ,$
```

INDEX=I is optional in the Master File, but using it is recommended so that keys in your Master File are easily recognized without reference to the Access File.

Describing Superdescriptors

A superdescriptor is a key value associated with all or part of two to twenty Adabas fields. If the superdescriptor consists of a group of complete fields, it is declared differently than a superdescriptor which consists of one or more partial fields. For information on superdescriptors consisting of partial fields, see *Customizing the Adabas Environment* on page 201.

Superdescriptors can be printed if they contain only alphabetic fields.

Specifying Superdescriptors Using the GROUP Attribute

A superdescriptor composed of several complete fields is described with the GROUP attribute. All the fields that are part of the group must be specified immediately after the GROUP attribute in the order in which they appear in the superdescriptor.

The following example shows a superdescriptor defined in server terminology:

```
GROUP=TRANS          ,ALIAS=S1 ,USAGE=A20   ,ACTUAL=A20 ,INDEX=I ,$
   FIELD=ACCT_NO     ,ALIAS=C2 ,USAGE=A9    ,ACTUAL=A9  ,$
   FIELD=ITEM_NO     ,ALIAS=GD ,USAGE=A5    ,ACTUAL=A5  ,$
   FIELD=TRANS_DATE ,ALIAS=D7 ,USAGE=A6YMD ,ACTUAL=A6  ,$
```

The group field name, in this case, is TRANS. Since it is a superdescriptor, the INDEX=I attribute is included on the GROUP specification.

The ALIAS is the two-character Adabas field name for the superdescriptor.

The USAGE and ACTUAL values must be specified in alphanumeric format. See *Calculating GROUP Length* on page 185 for more information.

If any of the components of the TRANS group field were descriptors, they could also contain the INDEX=I attribute.

Note: TYPE=SPR must be defined in the Access File. For more information, see *Customizing the Adabas Environment* on page 201.

Calculating GROUP Length

For a group field, you must supply both the USAGE and ACTUAL values in alphanumeric format.

If the component fields of the group are alphanumeric, the USAGE and ACTUAL lengths of the group field must be exactly the sum of the component field lengths. For example, in the *Specifying Superdescriptors Using the GROUP Attribute* on page 185, the lengths specified in the USAGE and ACTUAL attributes for the component fields ACCT_NO, ITEM_NO, and TRANS_DATE each total 20. The lengths specified in the USAGE and ACTUAL attributes of the TRANS group field are consequently both equal to 20.

If the component fields of the group are *not* alphanumeric:

- The USAGE and ACTUAL formats of the group field must still be alphanumeric.
- The ACTUAL length of the group field is still the sum of the component field lengths.
- The USAGE length is the sum of the internal storage lengths of the component fields.

You determine these internal storage lengths as follows:

USAGE	Length
A (alphanumeric)	Value equal to the number of characters contained in the field
D (decimal, double-precision)	8
F (decimal, single precision)	4
I (integer)	4
P (packed decimal):	
P*n* or P*n.d*, where *n* is less than or equal to 15	8
P16	16
P16.*d*	8
P*n* or P*n.d*, where *n* is greater than 16 or less than or equal to 31	16
Z (zoned)	Value equal to the number of characters contained in the field

For example, consider the following GROUP specification in the EMPLOYEES Master File:

```
GROUP=LEAVE_DATA    ,ALIAS=A3 ,A16 ,A4 ,$
  FIELD=LEAVE_DUE   ,ALIAS=AU ,P2  ,Z2 ,$
  FIELD=LEAVE_TAKEN ,ALIAS=AV ,P2  ,Z2 ,$
```

In this example:

- The lengths of the ACTUAL values for the component fields LEAVE_DUE and LEAVE_TAKEN total 4, which is the length of the ACTUAL value for the group field.

- The length of the USAGE value for the group field is determined by adding the internal storage lengths of the component fields LEAVE_DUE and LEAVE_TAKEN, as specified by the field types: a length of 8 for USAGE P2 (8 plus 8 = 16).

Here is a GROUP specification:

```
GROUP=MODEL_YEAR_MAKE ,ALIAS=AO ,A28 ,A22 ,INDEX=I ,$
  FIELD=YEAR          ,ALIAS=AG ,P2  ,Z2  ,$
  FIELD=MAKE          ,ALIAS=AD ,A20 ,A20 ,INDEX=I ,$
```

In this example:

- The lengths of the ACTUAL values for the component fields YEAR and MAKE total 22, which is the length for the ACTUAL value for the group field.
- The length of the USAGE value for the group field is determined by adding the internal storage lengths of the component fields YEAR and MAKE, as specified by the field types: a length of 8 for USAGE P2 and a length of 20 for USAGE A20 (8 plus 20 = 28).

Describing Descriptors in the Access File

Field suffixes are specified in the Access File to identify descriptors, superdescriptors, and subdescriptors.

The syntax is

```
FIELD[NAME]=fieldname, TYPE=fieldsuffix, [NU={YES|NO}] ,$
```

where:

fieldname

Is the field name or group in the Access File.

fieldsuffix

Indicates the field suffix. Possible values are:

DSC which indicates a descriptor.

SPR which indicates a superdescriptor made up of whole fields.

NOP which indicates a subdescriptor or superdescriptor made up of partial fields.

PDS which indicates a phonetic descriptor.

HDS which indicates a hyperdescriptor.

blank which indicates non-descriptor fields.

NU

Specifies whether null-suppression is in use. Possible values are:

YES which indicates that the field is described in the adapter with null-suppression.

Note: Descriptors with null values are not stored in inverted lists.

NO which indicates null-suppression is not used. NO is the default value.

Specifying Superdescriptors Containing Partial Fields

If a superdescriptor consists of one or more partial fields, that superdescriptor (field) is defined with TYPE=NOP (non-printable) in the Access File. This type of descriptor can be used in selection tests, but neither it nor any part of it can be printed or displayed unless the field is alphanumeric.

The partial fields that comprise such superdescriptors are stored in Adabas. There is no facility or necessity for identifying them to the server or the adapter. If you look at an Access File that contains a superdescriptor composed of partial fields, you would recognize it by TYPE=NOP, but you cannot list the partial fields that are its components. To use the adapter, identify such a superdescriptor in the Access File with TYPE=NOP.

Any superdescriptor defined to the server with TYPE=NOP has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

See your Software AG documentation for more information about using superdescriptors containing partial fields.

Specifying Subdescriptors

A subdescriptor consists of a partial field value for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=NOP. It can be used in selection tests, but it cannot be printed.

Any subdescriptor defined in the Access File with TYPE=NOP has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

Specifying Phonetic Descriptors

A phonetic descriptor consists of similar phonetic values for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=PDS. It can be used in selection tests, but it cannot be printed.

Specifying Hyperdescriptors

A hyperdescriptor consists of values generated, based on a user-supplied algorithm, for which an index has been created in Adabas. It is described at the field level in the Access File as TYPE=HDS. It can be used in selection tests, but it cannot be printed.

Any phonetic descriptor or hyperdescriptor defined in the Access File with TYPE=PDS/HDS has the following limitation: CALLTYPE=RL is not supported when this field is the IXFLD in an embedded cross-reference or JOIN.

If you have selected a field defined with TYPE=NOP (for example, subdescriptors), TYPE=PDS or TYPE=HDS, a FIND call is issued even if RL has been specified.

Specifying Null-Suppression

To allow the Adapter for Adabas to create the most efficient calls, while still maintaining integrity of the answer set, any null-suppressed fields that are components of a superdescriptor must be defined in the Access File. The NU attribute is used to define a null-suppressed field.

An NU field defined as a descriptor is not stored in inverted lists when it contains a null value. Any qualifying descriptor records containing a null value are not recognized by a FIND command that refers to that descriptor.

The same is true for subdescriptors and superdescriptors derived from fields described to the adapter with null-suppression. No entry is made for a subdescriptor if the bytes of the field from which it is derived contain a null value and that field is defined with null-suppression (NU). No entry is made for a superdescriptor if any of the fields from which it is derived is an NU field with a null value.

For more information about null-suppression and how it affects data retrieval, see your Software AG documentation.

Specifying Support for SQL-Compatible Null Representation

Two additional data definition options are provided to enable specification of SQL-compatible null representation and handling by the Adapter for Adabas. The NC (not counted) and NN (not null) options can be placed in Access File field definitions to control evaluation and handling of null values. (Note that the NN option is only permitted when the NC option is also specified.)

See *Server File Structure* on page 143 for an overview of these options, and *Creating Synonyms When Adabas Fields Are Defined With the NC Option* on page 152 for a detailed explanation and examples of both the Master Files and Access Files following issuance of Create Synonym.

Implementing Embedded JOINs: KEYFLD and IXFLD

The KEYFLD and IXFLD attributes identify the common fields for parent/descendant relationships in a multi-segment Master File. These relationships are referred to as embedded JOINs or server views.

For each descendant segment, the KEYFLD and IXFLD attributes specify the field names of the shared field that implements the embedded JOIN. The parent field supplies the value for cross-referencing; the descendant field contains the corresponding value. The adapter implements the relationship by matching values at run time.

The KEYFLD and IXFLD attributes are valid only for ACCESS=ADBS segments.

Attribute	Description
KEYFLD	Field name in the parent segment whose value is used to retrieve the child segment. This is also known as the primary key.
IXFLD	Field name in the child or cross-referenced segment containing the related data. This is also known as the foreign key.

The value for the KEYFLD attribute is a 1- to 66-character field name or alias from the parent segment. The value for the IXFLD attribute is a 1- to 66-character field name or alias from the descendant segment. This is an example of a Master File with an embedded JOIN using KEYFLD and IXFLD.

```
FILENAME=AMKTORDR     ,SUFFIX=ADBSINX ,$
SEGNAME=MKTORDER
FIELDNAME=NMARKET_GRP     ,BA          ,I3      ,I2,   INDEX=I,$
FIELDNAME=QPRODUCT        ,BB          ,I3      ,I2       ,$
FIELDNAME=QNEEDITM        ,BC          ,A3      ,I2       ,$
FIELDNAME=FBUILD          ,BD          ,A1      ,A1       ,$
FIELDNAME=FK_NPRODUCT     ,BE          ,A4      ,A4       ,$
FIELDNAME=FK_NCUSTOMER    ,BF          ,I3      ,I2       ,$
FIELDNAME=DATEMKTO        ,BG          ,A8      ,A8       ,$
FIELDNAME=DATEFBLD        ,BH          ,A8      ,A8       ,$

SEGNAME=CUSTOMER, SEGTYPE=U,PARENT=MKTORDER,$
FIELDNAME=NCUSTMR_GRP     ,AA          ,I3      ,I2,   INDEX=I,$
FIELDNAME=NAMECUST        ,AB          ,A15     ,A15      ,$
FIELDNAME=DCUSROAD        ,AC          ,A20     ,A20      ,$
FIELDNAME=DCUSTOWN        ,AD          ,A20     ,A20      ,$
```

This is an example of an Access File with an embedded JOIN using KEYFLD and IXFLD.

```
RELEASE=6,    OPEN=YES,$
  SEGNAM=MKTORDER,    ACCESS=ADBS,FILENO=022,DBNO=001 ,$
  SEGNAM=CUSTOMER,    ACCESS=ADBS,FILENO=021,DBNO=001 ,CALLTYPE=FIND,
IXFLD=NCUSTMR_GRP,KEYFLD=FK_NCUSTOMER,$
```

Note: Include the pair of attributes in the Access File segment declaration for descendant segments. Do not specify them in the segment declaration for the root segment.

A JOIN can be based on more than one field in the host and cross-referenced logical record types. If the Adabas file uses multiple fields to establish a relationship or link between logical record types, you can specify concatenated fields in an embedded JOIN. You can also specify multiple fields with the dynamic JOIN command.

In the multi-field embedded JOIN, the KEYFLD and IXFLD values consist of a list of their component fields separated by slashes (/). Additional Access File attributes are not required. The syntax is

```
KEYFLD=field1/field2/...,
IXFLD=cfield1/cfield2/...,$
```

where:

field1/field2...

Is a composite of up to 16 key fields from the parent segment. Slashes are required.

cfield1/cfield2...

Is a composite of up to 16 key fields from the descendant segment.

The adapter compares each field pair for the data formats prior to format conversion. It evaluates each field pair with the following rules:

- The cross-referenced field in the embedded JOIN must be an Adabas descriptor field. The host field can be any field.
- Parent and descendant fields must be real fields.
- All participating fields for either the parent or descendant file must reside in the same segment.
- KEYFLD and IXFLD attributes must specify the same number of fields. If the format of the parent field is longer than the format of the descendant field, its values are truncated. If the format of the parent field is shorter, its values are padded with zeros or blanks.
- The KEYFLD and IXFLD attributes can consist of a maximum of 16 concatenated fields in order of high-to-low significance. You must specify the field order: the order determines how the values will be matched.
- The pair of fields in the KEYFLD/IXFLD attribute does not have to have comparable data formats.

To implement JOINs, the Adabas DBMS converts the alphanumeric field formats on the server to equivalent Adabas field formats in order to perform the necessary search and match operations. When the Adabas DBMS returns the answer set of matched values, it also converts the values back to alphanumeric formats. The cross-referenced fields must be descriptor fields.

Mapping Adabas Files With Variable-Length Records and Repeating Fields

The OCCURS segment attribute is used by the server to describe MU and PE field types in a Master File.

Consider the sample Adabas file VEHICLES. The FDT for the VEHICLES file is shown below.

```
**********************************
* FILE     2 (VEHICLES        ) *
**********************************
FIELD DESCRIPTION TABLE
      I      I        I        I              I                    I
LEVEL I NAME I LENGTH I FORMAT I  OPTIONS     I  PARENT OF         I
      I      I        I        I              I                    I
------I------I--------I--------I--------------I--------------------I
      I      I        I        I              I                    I
  1   I  AA  I   15   I    A   I DE,UQ,NU     I                    I
  1   I  AB  I    4   I    F   I FI           I                    I
  1   I  AC  I    8   I    A   I DE           I                    I
  1   I  CD  I        I        I              I                    I
  2   I  AD  I   20   I    A   I DE,NU        I SUPERDE            I
  2   I  AE  I   20   I    A   I NU           I                    I
  2   I  AF  I   10   I    A   I DE,NU        I                    I
  1   I  AG  I    2   I    U   I NU           I SUPERDE            I
  1   I  AH  I    1   I    A   I DE,FI        I                    I
  1   I  AI  I    1   I    A   I FI           I                    I
  1   I  AJ  I    6   I    U   I NU           I SUPERDE            I
  1   I  AK  I        I        I              I                    I
  2   I  AL  I    3   I    A   I NU           I                    I
  2   I  AM  I    4   I    P   I MU,NU        I                    I
      I      I        I        I              I                    I
-------------------------------------------------------------------
SPECIAL DESCRIPTOR TABLE
       I      I        I        I                I                 I
 TYPE  I NAME I LENGTH I FORMAT I    OPTIONS     I   STRUCTURE     I
       I      I        I        I                I                 I
-------I------I--------I--------I----------------I-----------------I
       I      I        I                         I                 I
 SUPER I  AN  I    4   I    B   I DE,NU          I AJ (  5 -   6) I
       I      I        I        I                I AJ (  3 -   4) I
 SUPER I  AO  I   22   I    A   I DE,NU          I AG (  1 -   2) I
       I      I        I        I                I AD (  1 -  20) I
       I      I        I        I                I                 I
-------------------------------------------------------------------
```

The MU field, MAINT_COST, is a field in VEHICLES. It is allocated to a new segment, AM0101, required by the adapter, as shown in the Master File for a partial view of the VEHICLES file, illustrating the use of OCCURS=*occursname*:

```
$$$ CREATED ON 12/10/97 AT 10.17.27 BY PMSMJB
FILENAME=ADACAR,SUFFIX=ADBSINX,$

$ ADABAS FILE = VEHICLES-FILE                    DICTIONARY = 6
SEGNAME=S01     ,SEGTYPE=S,$

 FIELD=REG_NUM                  ,ALIAS=AA      ,A15 ,A15  ,INDEX=I,$
 FIELD=CHASSIS_NUM              ,ALIAS=AB      ,I9  ,I4   ,$
 FIELD=PERSONNEL_ID             ,ALIAS=AC      ,A8  ,A8   ,INDEX=I,$
 FIELD=CLASS                    ,ALIAS=AH      ,A1  ,A1   ,INDEX=I,$
 FIELD=LEASE_PUR                ,ALIAS=AI      ,A1  ,A1   ,$
 FIELD=DATE_ACQ                 ,ALIAS=AJ      ,P6  ,Z6   ,$
$GRMU=CAR_MAINTENANCE           ,ALIAS=AK      ,A11 ,A7   ,$
  FIELD=CURR_CODE               ,ALIAS=AL      ,A3  ,A3   ,$
  FIELD=MAINT_COST_CNT          ,ALIAS=AMC     ,I4  ,I2   ,$
 FIELD=DAT_ACQ_DESC             ,ALIAS=AN      ,A4  ,A4   ,INDEX=I,$
 GROUP=MODEL_YEAR_MAKE          ,ALIAS=AO      ,A28 ,A22  ,INDEX=I,$
  FIELD=YEAR_S02                ,ALIAS=AG      ,P2  ,Z2   ,$
  FIELD=MAKE_S02                ,ALIAS=AD      ,A20 ,A20  ,INDEX=I,$

SEGNAME=AM0101 ,SEGTYPE=S,PARENT=S01  ,OCCURS=AMC,$  MAX= 60
 FIELD=MAINT_COST               ,ALIAS=AM      ,P7  ,P4   ,$
 FIELD=AM0101_OCC               ,ALIAS=ORDER   ,I4  ,I2   ,$
```

In the original segment, the MAINT_COST_CNT (ALIAS=AMC) field contains the total number of occurrences of the MAINT_COST field. The new segment, AM0101, contains the MAINT_COST data field and the newly created AM0101_OCC field. The AM0101_OCC field contains the position of each occurrence.

This example includes the group field called CAR_MAINTENANCE, which contains the MU field, MAINT_COST. You cannot report on a group with an unknown length. However, the component fields can be referenced for reporting purposes.

Note: The Create Synonym facility creates the new segment, AM0101, for you. Otherwise, you must create the new segment using the OCCURS segment attribute.

The following is the Access File for the VEHICLES file:

```
$$$ CREATED ON 12/10/03 AT 10.17.27 BY PMSMJB
$$$ FILENAME=ADACAR,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = VEHICLES-FILE                      DICTIONARY = 6
SEGNAM=S01   ,ACCESS=ADBS,FILENO=002,
              CALLTYPE=RL,SEQFIELD=PERSONNEL_ID,$
 FIELD=DAT_ACQ_DESC                          ,TYPE=NOP,$
 FIELD=MODEL_YEAR_MAKE                       ,TYPE=SPR,$
  FIELD=YEAR_S02                             ,TYPE=   ,NU=YES,$
  FIELD=MAKE_S02                             ,TYPE=DSC,NU=YES,$
SEGNAM=AM0101,ACCESS=MU  ,FILENO=002,$ MAINT_COST
```

A new segment is created for the MU field. Note that for this new segment, ACCESS=MU.

Here is an example of the counter fields. The AMC field contains the total count of occurrences, the AM field contains the actual data, and the ORDER field contains the position of each occurrence.

```
     (AMC)          (AM)        (ORDER)
MAINT_COST_CNT   MAINT_COST   AM0101_OCC
--------------   ----------   ----------
             1          520            1
-----------------------------------------
             1          210            1
-----------------------------------------
             4           44            1
             4          322            2
             4           66            3
             4          188            4
-----------------------------------------
             6          324            1
             6         1103            2
             6          566            3
             6          755            4
             6          988            5
             6         1899            6
-----------------------------------------
             2          344            1
             2          500            2
```

In some cases, the ORDER field may describe the months of the year (1 to 12). If all occurrences for the month of June are needed, for example, you would print data in which the ORDER field is equal to 6 (for June).

Using the preceding sample data, here is an example of how to use the ORDER field. This request always selects the sixth occurrence of maintenance costs:

```
TABLE FILE VEHICLES
PRINT MAINT_COST
WHERE AM0101_OCC EQ 6
END

MAINT_COST
----------
      1899
```

Using the same data, this request always selects the second occurrence of maintenance costs:

```
TABLE FILE VEHICLES
PRINT MAINT_COST
WHERE AM0101_OCC EQ 2
END

MAINT_COST
----------
       322
      1103
       500
```

Describing Multi-Value Fields and Periodic Groups With the OCCURS Attribute

The OCCURS attribute is a segment attribute used to describe portions of records containing a multi-value field or a periodic group. When describing Adabas files, the OCCURS attribute may be equal to the field name or (another option) to the two-character internal Adabas name appended with the letter C.

Records with repeating fields are defined in the Master File by describing the singly occurring fields in the parent segment, and the multiply occurring fields or groups in their own subordinate segment. The OCCURS attribute appears in the declaration for the subordinate segments.

The syntax is

```
OCCURS=occursname ,$
```

where:

occursname

Indicates a data field in the parent segment that contains the number of occurrences of the descendant segment. Its value is derived from the two-character internal Adabas name (ALIAS attribute in the Master File) appended with the letter C.

The following is an example of the VEHICLES Master File, containing an MU segment with the counter field in the parent segment:

```
FILENAME=VEHICLES,SUFFIX=ADBSINX,$
  SEGNAME=S01     ,SEGTYPE=S,$
   FIELD=REG_NUM          ,ALIAS=AA          ,A15    ,A15  , INDEX=I,$
   FIELD=CHASSIS_NUM      ,ALIAS=AB          ,I9     ,I4   ,$
   FIELD=PERSONNEL_ID     ,ALIAS=AC          ,A8     ,A8   , INDEX=I,$
   FIELD=CLASS            ,ALIAS=AH          ,A1     ,A1   , INDEX=I,$
   FIELD=LEASE_PUR        ,ALIAS=AI          ,A1     ,A1   ,$
   FIELD=CURR_CODE        ,ALIAS=AL          ,A3     ,A3   ,$
   FIELD=MAINT_COST_CNT   ,ALIAS=AMC         ,I4     ,I2   ,$
   GROUP=MODEL_YEAR_MAKE  ,ALIAS=AO          ,A28    ,A22  , INDEX=I,$
    FIELD=YEAR            ,ALIAS=AG          ,P2     ,Z2   ,$
    FIELD=MAKE            ,ALIAS=AD          ,A20    ,A20  , INDEX=I,$
SEGNAME=AM0101  ,SEGTYPE=S ,PARENT=S01  ,OCCURS=AMC,$
   FIELD=MAINT_COST       ,ALIAS=AM          ,P7     ,P4   ,$
   FIELD=AM0101_OCC       ,ALIAS=ORDER       ,I4     ,I2   ,$
```

MAINT_COST is the MU field. It is allocated to the newly created segment, AM0101. The counter field, MAINT_COST_CNT, is located in the parent segment. The new segment, AM0101, uses this counter field to determine the number of occurrences of the MAINT_COST field (OCCURS=AMC).

Notice that the fields MAINT_COST_CNT and AM0101_OCC (ALIAS=ORDER) have USAGE=I4 and ACTUAL=I2. These are the required values for the USAGE and ACTUAL attributes of the counter field and the ORDER field.

The adapter builds the PE segment in the same way. However, the component fields within the PE group are defined in the new segment.

The following is a view of the EMPLOYEES Master File, containing a PE group:

```
FILENAME=EMPFILE1,SUFFIX=ADBSINX,$

SEGNAME=S01     ,SEGTYPE=S,$
 FIELD=PERSONNEL_ID              ,ALIAS=AA       ,A8    ,A8  ,INDEX=I,$
  FIELD=FIRST_NAME               ,ALIAS=AC       ,A20   ,A20 ,$
  FIELD=NAME                     ,ALIAS=AE       ,A20   ,A20 ,INDEX=I,$
FIELD= LEAVE_BOOKED_CNT          ,ALIAS=AWC      ,I4    ,I2  ,$
SEGNAME=AW0401   ,SEGTYPE=S,PARENT=S01   ,OCCURS=AWC,$
 GROUP=LEAVE_BOOKED              ,ALIAS=AW       ,A16   ,A12 ,$
  FIELD=LEAVE_START              ,ALIAS=AX       ,P6    ,Z6  ,$
  FIELD=LEAVE_END                ,ALIAS=AY       ,P6    ,Z6  ,$
 FIELD=AW0401_OCC                ,ALIAS=ORDER    ,I4    ,I2  ,$
```

LEAVE_BOOKED is the PE group. It is allocated to the newly created segment, AW0401. The counter field, LEAVE_BOOKED_CNT, is located in the parent segment. The new segment, AW0401, uses this counter field to determine the number of occurrences of the LEAVE_BOOKED group (OCCURS=AWC).

The counter field, LEAVE_BOOKED_CNT, and the ORDER field, AW0401_OCC, require values of I4 for the USAGE attribute and I2 for the ACTUAL attribute.

The number of occurrences is limited by the original definition of the file to Adabas.

To retrieve the appropriate occurrence of a repeating field in a report request, specify the ORDER field in your Master File. See *Specifying OCCURS Segment Sequence: The ORDER Field* on page 198 for details.

For detailed information about defining MU fields and PE groups in the Access File, see *Segment Attributes in Access Files* on page 177.

Describing Multi-Value Fields Within Periodic Groups

This topic describes how to specify multi-value fields within periodic groups in the Master File. Define each multi-value field contained in the periodic group segment as a separate descendant segment of the periodic group.

A periodic group containing a multi-value field is defined as two separate segments.

All fields must be defined in the order in which they appear in the FDT.

Consider the EMPLOYEES file. It has a periodic group called INCOME, which contains one multi-value field, BONUS. Each counter field (for example, AQC) must be in its appropriate parent segment. The INCOME_CNT field is in the parent segment for the periodic group, INCOME. The BONUS_CNT field is in the parent segment for the MU field, BONUS. Any repeating fields within the periodic group are defined in the AQ0201 segment. Their corresponding Adabas two-character field names are the values for ALIAS with the appropriate values for USAGE and ACTUAL.

Here is the view of the Master File for this structure:

```
 FILENAME=EMPFILE1,SUFFIX=ADBSINX,$

   SEGNAME=S01     ,SEGTYPE=S,$
    FIELD=PERSONNEL_ID      ,ALIAS=AA        ,A8    ,A8  ,INDEX=I,$
     FIELD=FIRST_NAME       ,ALIAS=AC        ,A20   ,A20 ,$
     FIELD=NAME             ,ALIAS=AE        ,A20   ,A20 ,INDEX=I,$
1. FIELD=INCOME_CNT        ,ALIAS=AQC       ,I4    ,I2  ,$

1. SEGNAME=AQ0201  ,SEGTYPE=S,PARENT=S01   ,OCCURS=AQC,$
1. $PEMU=INCOME            ,ALIAS=AQ        ,A19   ,A13 ,$
1.    FIELD=CURR_CODE       ,ALIAS=AR        ,A3    ,A3  ,$
1.    FIELD=SALARY          ,ALIAS=AS        ,P9    ,P5  ,$
2.   FIELD=BONUS_CNT        ,ALIAS=ATC       ,I4    ,I2  ,$
1. FIELD=AQ0201_OCC        ,ALIAS=ORDER     ,I4    ,I2  ,$

2. SEGNAME=AT0301  ,SEGTYPE=S,PARENT=AQ0201,OCCURS=ATC,$
2.  FIELD=BONUS            ,ALIAS=AT        ,P9    ,P5  ,$
2.  FIELD=AT0301_OCC       ,ALIAS=ORDER     ,I4    ,I2  ,$
```

1. PE fields.

This example includes the periodic group called INCOME, which contains the MU field, BONUS. Create Synonym comments out the group field with $PEMU because its length is variable (depending on the number of occurrences of the MU field). The server cannot report on a group with an unknown length. However, the component fields can be referenced for reporting purposes.

2. MU fields.

   BONUS_CNT (ALIAS=ATC) is the counter field. It is defined in the PE segment and tells us the number of occurrences for that field. Used in the MU segment, it indicates how many times this repeating field occurs, for example, OCCURS=ATC. Defined in the parent segment, the counter field is used by the child segment to tell us how many occurrences to expect.

   Any periodic group segment must be described in its entirety. This description has two requirements: each field of the periodic group must have a field declaration in the periodic group segment, and each field in the group must be described in the order in which it appears in the Adabas FDT.

Specifying OCCURS Segment Sequence: The ORDER Field

There may be occasions with OCCURS segment processing when the order of the data is significant. For example, the field values may represent monthly or quarterly data, but the record itself may not explicitly specify the month (or quarter) to which the data applies. The ORDER field maintains the sequence number for each multiply occurring instance.

The order of the occurrences of a multi-value field or periodic group also has some significance in your application. For example, if a periodic group contains 12 occurrences, each occurrence could correspond to one month of the year. The first occurrence represents January, the second February, and so on through to December.

To use the ORDER option, you must include an additional field in your Master File with an ALIAS of ORDER. This option instructs the server to determine the sequence number of fields in an OCCURS segment. The server automatically supplies a value for the new field that defines the sequence number of each repeating group. The ORDER field does not represent an existing Adabas field; it is used only for internal processing.

The syntax rules for the ORDER field are:

- It must be the last field described in the OCCURS segment.
- The field name must be unique.
- The ALIAS value must be ORDER.

- The USAGE value must be I4, with any appropriate display options.
- The ACTUAL value must be I2 (in earlier releases, I1 was used).

Each field with the ALIAS value ORDER tracks the order of occurrence of the preceding fields in the segment. Therefore, in the sample Master File in *Describing Multi-Value Fields Within Periodic Groups* on page 197, AT0301_OCC maintains the sequence of BONUS occurrences.

An ORDER field may be decoded into a meaningful value in a DEFINE.

Using the GROUP Attribute to Cross-Reference Files

The GROUP attribute can be used to cross-reference Adabas files if the file you want to search does not contain a descriptor. This cross-referencing can be done *only* if fields within the file you want to search correspond to descriptors in the file you are cross-referencing.

Consider this situation: You have a SALES file which contains salesman information. It also has account information, such as COMPANY_CODE, ITEM_NUMBER, and ITEM_NAME.

Suppose you also have an ACCOUNT file containing information about each company in the territory of a salesman. The ACCOUNT file has a superdescriptor consisting of COMP_CODE, IT_NUMBER, and IT_NAME. These fields correspond to the fields specified in the SALES file.

You can create a link between the matching fields in the SALES and ACCOUNT files. To do this, describe the matching fields from the SALES file as a group. Then you can join the group field to the superdescriptor in the ACCOUNTS file. The host file is the file, without a descriptor, containing the held values which must be grouped in order to retrieve related records in the second file. For example:

```
FILE=SALES     ,SUFFIX=ADBSINX     ,$
  SEGNAME=ACCT_SEG  ,SEGTYPE=S    ,$
    GROUP=LINKFLDS          ,ALIAS=    ,USAGE=A20 ,ACTUAL=A20 ,$
      FIELD=COMPANY_CODE    ,ALIAS=BA ,USAGE=A3  ,ACTUAL=A3  ,$
      FIELD=ITEM_NUMBER     ,ALIAS=BB ,USAGE=A5  ,ACTUAL=A5  ,$
      FIELD-ITEM_NAME       ,ALIAS=BC ,USAGE=A12 ,ACTUAL=A12 ,$
```

SALES is the host file which uses the superdescriptor (COMPANY) in the ACCOUNT file for the company-related data. The superdescriptor in the ACCOUNT file, easily recognized by TYPE=SPR in the Access File, is composed of fields that correspond to the dummy group in the host. Looking at the GROUP attribute used to cross-reference files, above, and the GROUP attribute used to cross-reference files, below, you see how the fields in the LINKFLDS group in the SALES file relate to the superdescriptor COMPANY in the ACCOUNT file:

```
FILE=ACCOUNT    ,SUFFIX=ADBSINX ,$
  SEGNAME=PROD_SEG  ,SEGTYPE=S  ,PARENT=ACCTS ,$
    GROUP=COMPANY     ,ALIAS=S1 ,USAGE=A20 ,ACTUAL=A20, INDEX=I,$
      FIELD=COMP_CODE ,ALIAS=AA ,USAGE=A3  ,ACTUAL=A3  ,$
      FIELD=IT_NUMBER ,ALIAS=AB ,USAGE=A5  ,ACTUAL=A5  ,$
      FIELD-IT_NAME   ,ALIAS=AC ,USAGE=A12 ,ACTUAL=A12 ,$
```

Use the GROUP superdescriptor COMPANY to retrieve data for those values using the JOIN command within a procedure. The JOIN command or embedded cross-reference completes the link. In this example, you would join LINKFLDS in SALES to COMPANY in ACCOUNT using the following syntax:

```
JOIN LINKFLDS IN SALES TO ALL COMPANY IN ACCOUNT AS J1
```

Platform-Specific Functionality

The Adapter for Adabas on UNIX and Windows, and OpenVMS, has more restrictions than on z/OS. These restrictions are described below.

On all platforms, if an error occurs while Adabas commands are being processed, then the communication block (CB) is returned along with the Response Code (RC) Additions 2 field, in 2 or 4 bytes binary format. The Additions 2 field indicates the specific reason for the Response Code. The Adapter for Adabas displays both RC and Additions 2 fields in the message. This information is available in the trace as well.

Reference: Functionality on z/OS

The following conditions apply when using the adapter on z/OS platforms:

- Sub/super descriptors defined with "NC=YES" and "MISSING=ON" parameters cannot be used to search for SQL NULL values, as that causes an Adabas RC 61.
- If you assign Server or Application Administrator privileges to user IDs other than the one used to install the server, to allow these additional administrators to create metadata for Adabas data sources, you must do one of the following:
  - Include the Adabas load library in your SYS1.PARMLIB(LNKLST*nn*) member.
  - Use the STEPLIBLIST feature and add the Adabas load library to the sanctioned list file. For full details on the STEPLIBLIST feature, see the IBM publication *UNIX System Services Planning Guide*.

    Tip: To locate the information in that document, search for STEPLIBLIST.

Reference: Functionality on UNIX and Windows

The following conditions apply when using the adapter on UNIX and Windows platforms:

- You cannot use non-descriptor fields as search criteria.
- The parameter NDFIND is automatically set to the OFF value if the adapter is executed in a non-mainframe environment. If non-descriptor fields are used as key fields, the entire database is read.
- If the Master File non-descriptor field is declared as an indexed field (INDEX=I or FIELDTYPE=I) and used as a key field, the result from Adabas is RC 61.
- If an occurrence of an MU (multi-value) field is deleted, the adapter shifts all other occurrences into its place and assigns an empty value for the last occurrence. If this field has an NU (null suppression) option, the last occurrence will be automatically deleted by Adabas.
- DBNO must be assigned to a valid numeric value for the existing database.

Reference: Functionality for OpenVMS Platform

Adabas on OpenVMS does not allow a direct request for sub/superdescriptors values. For this reason, there are some limitations on using these fields (with TYPE=NOP/SPR in ACX) in server requests to Adabas:

- NOP fields cannot be printed or displayed, even if they are alphanumeric.
- NOP/SPR fields cannot be used in functions (MIN, MAX, and so on).
- If NOP/SPR fields are used in the IF/WHERE criteria, then CALLTYPE=RL is automatically changed to CALLTYPE=FIND. It can change the sequence of returning data rows on OpenVMS in comparison with sequence on other platforms.
- NOP fields cannot be used in a complex IF/WHERE clause.
- NOP fields cannot be used for joining.

Customizing the Adabas Environment

The Adapter for Adabas provides several parameters for customizing the environment and optimizing performance.

The sections *Multifetch and Prefetch* on page 205, and *Adabas Dynamic Database Number* on page 207 describe the environment commands that display and change the parameters governing your server session.

ORDER Fields in the Indexed Field List

ORDER fields maintain the sequence number for each multiply occurring instance. ON is the default value and includes ORDER fields in the list of indexed fields.

Setting ORDERKEYS to OFF forces the Adapter for Adabas to exclude ORDER fields from the list of indexed fields.

Syntax: How to Set ORDERKEYS in the Indexed Field List

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET ORDERKEYS {ON|OFF}
```

where:

`ON`

Includes ORDER fields in the list of indexed fields. ON is the default value.

`OFF`

Excludes ORDER fields from the list of indexed fields.

Switching the Access Mode Using NOPACCESS

This setting allows you to switch the access mode from RL to FIND if a field defined with TYPE=NOP was selected (even if CALLTYPE=RL has been specified).

The default value for this setting is MIXED, which uses the default access mode for the selected field. The new setting can be useful when the NOP superdescriptor defined in a root segment is derived from fields that belong to different Master File segments.

Syntax: How to Set NOPACCESS

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET NOPACCESS {FIND|MIXED}
```

where:

FIND

Switches the access mode from RL to FIND if a field defined with TYPE=NOP was selected (even if CALLTYPE=RL has been specified).

MIXED

Uses the default access mode for the selected field.

NEW Synonym Setting for Superdescriptors

Setting SYNONYM to NEW instructs the Adapter for Adabas to use new logic when:

- Describing metadata.
- Processing requests that can take advantage of superdescriptors for screening conditions.

Note: This command must be specified in the server profile.

Syntax: How to Set Synonyms for Superdescriptors

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET SYNONYM {NEW|STD}
```

where:

NEW

Describes superdescriptors as groups in the Master File. With NEW synonym format, neither the groups nor their components (fields) have field names specified in the Master File. Aliases are populated using the Adabas FDT (2-byte name).

When a synonym contains the NEW syntax, the Adapter for Adabas uses NEW logic to process the selection criteria of requests involving that synonym. Screening conditions within the request are analyzed and the superdescriptor that covers the highest order component fields required by the selection criteria are used.

Note that you must specify CALLTYPE=RL in the Access File in order to take advantage of superdescriptor-based access.

STD

Incorporates the usual synonym creation behavior, which does not trigger the new logic for requests using superdescriptors. STD is the default value.

For more information on synonyms, see *Creating Synonyms* on page 147.

Example: Setting Synonyms for Superdescriptors

The following Master File illustrates superdescriptors Y6 and Y7 defined as GROUP, with the NEW synonym format:

```
GROUP=      ,ALIAS=Y6          ,A2    ,A2    ,INDEX=I,$
   FIELD=   ,ALIAS=AF          ,A1    ,A1    ,       ,$
   FIELD=   ,ALIAS=AG          ,A1    ,A1    ,       ,$
GROUP=      ,ALIAS=Y7          ,A22   ,A22   ,INDEX=I,$
   FIELD=   ,ALIAS=AF          ,A1    ,A1    ,       ,$
   FIELD=   ,ALIAS=AG          ,A1    ,A1    ,       ,$
   FIELD=   ,ALIAS=AC          ,A20   ,A20   ,       ,$
```

The corresponding Access File specifies the superdescriptors Y6 and Y7 and their components, using the keywords SUPER and NUMFLDS:

```
SUPER=Y6                ,NUMFLDS=2          ,$
  FIELD=AF_FIELD        ,TYPE=   ,NU=NO     ,$
  FIELD=AG_FIELD        ,TYPE=   ,NU=NO     ,$
 SUPER=Y7                ,NUMFLDS=3          ,$
  FIELD=AF_FIELD        ,TYPE=   ,NU=NO     ,$
  FIELD=AG_FIELD        ,TYPE=   ,NU=NO     ,$
  FIELD=AC_FIELD        ,TYPE=   ,NU=NO     ,$
```

SUPER specifies the native Adabas two-byte field name of the superdescriptor.

NUMFLDS defines the number of participating fields. Descriptions of all participating fields immediately follow the SUPER statement.

The request that follows uses superdescriptor Y6 in the L3 (Read Logical) command in the Access File:

```
TABLE FILE EMP211
PRINT AF_FIELD AG_FIELD AC_FIELD AD_FIELD
IF AF_FIELD EQ 'S'
IF AG_FIELD EQ 'M'
END
```

The next request uses superdescriptor Y7 in the L3 (Read Logical) command. Notice that it references an extra field in the last line of the Access File:

```
TABLE FILE EMP211
PRINT AF_FIELD AG_FIELD AC_FIELD AD_FIELD
IF AF_FIELD EQ 'S'
IF AG_FIELD EQ 'M'
IF AC_FIELD EQ 'C'
END
```

Multifetch and Prefetch

The Adabas Multifetch and Prefetch options reduce execution time and allow Adabas data to be retrieved with a high degree of efficiency. By buffering multiple record results from a single call, Multifetch and Prefetch reduce the communication overhead between the Adapter for Adabas and the Adabas nucleus.

The adapter uses the Multifetch option if it is available (this is the default option), or the Prefetch option if it is available. The option available is determined by your environment.

The adapter trace file contains the following information about the Fetch option:

- Fetch is allowed or disallowed.
- Fetch technique being used (Multifetch or Prefetch).
- Number of records per Fetch buffer.
- Size of the Fetch buffer (ISN buffer).

The Multifetch and Prefetch options require that OPEN=YES (the default value) is specified in the Access File. When OPEN=YES is specified, the Multifetch and Prefetch options are activated. There are two ways to deactivate (or reactivate) these options:

- Issue SET commands before running the request.
- Include the FETCH and FETCHSIZE attributes in the Access File.

Note: The SET commands override the Access File settings. This setting will be in effect for the entire server session.

Syntax: How to Set FETCH and FETCHSIZE

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET FETCH {ON|OFF|DEFAULT}
ENGINE ADBSINX SET FETCHSIZE {n|MAX[IMUM]}
```

where:

`ADBSINX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Sets the Fetch feature on for the user session.

OFF

Sets the Fetch feature off for the user session.

DEFAULT

Uses the information from the Access File. DEFAULT is the default value.

n

Is the number of records per buffer (1–32K). The buffer size varies with the size of the record and can be different for each TABLE request. The buffer size limit is 32K. 10 is the default value.

MAX[IMUM]

Is automatically calculated by the Adapter for Adabas to allow the maximum number of records that fit in a 32K buffer.

Syntax: How to Declare the FETCH and FETCHSIZE Attributes in the Access File

FETCH and FETCHSIZE are applicable only to segments described as ACCESS=ADBS in the Access File.

The Access File can contain the following attributes in the SEGNAM statement

```
FETCH={ON|OFF}
FETCHSIZE={n|MAX[IMUM]}
```

where:

ON

Sets the Fetch feature on for the user session. ON is the default value.

OFF

Sets the Fetch feature off for the user session.

n

Is the number of records per buffer (1 to 32k). 10 is the default value.

MAX[IMUM]

Is automatically calculated by the Adapter for Adabas to allow the maximum number of records that fit in a 32K buffer.

Here is an example of the FETCH and FETCHSIZE attributes in the sample Access File for EMPLOYEES:

```
$$$ FILENAME=EMPFILE1,SUFFIX=ADBSINX,$
RELEASE=6,OPEN=YES,$

$ ADABAS FILE = EMPLOYEES                          DICTIONARY =
SEGNAM=S01 ,ACCESS=ADBS,FILENO=001,CALLTYPE=RL,
             SEQFIELD=PERSONNEL_ID,FETCH=ON,FETCHSIZE=15,$
FIELD=DEPT_PERSON         ,TYPE=SPR,$
FIELD=DEPT_S03            ,TYPE=DSC,NU=YES,$
FIELD=NAME_S03            ,TYPE=    ,NU=NO,$
```

See your Software AG documentation for more information about the Multifetch/Prefetch feature.

Controlling Adabas Multifetch for Join Operations

You can use the FETCHJOIN command to take advantage of Multifetch efficiencies in Join operations for a session. This feature is particularly useful when you are joining to a base file in which sorting is based on the key field that is used for the join.

Syntax: How to Set Adabas FETCHJOIN

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET FETCHJOIN {ON|OFF}
```

where:

`ADBSINX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Sets the FETCHJOIN feature on for the user session. ON is the default value.

`OFF`

Sets the FETCHJOIN feature off for the user session. This value is recommended when joining with a base file that is unsorted by the key field used for joining. This option avoids buffering and reduces processing time.

Adabas Dynamic Database Number

Previous versions of the Adapter for Adabas required specification of the Adabas database number in the Access File using the DBNO attribute. This feature required a duplicate copy of the Access File for each database accessed.

Adabas database numbers can now be set from the server's profile. This SET command allows users to override the DBNO in the Access File. Note that specifying database numbers in the Access File is still supported. Use of the dynamic database number makes Master and Access Files shareable among databases.

The SET command remains valid throughout the server session.

Syntax: How to Set the Adabas Dynamic Database Number

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ADBSINX SET DBNO dbno
ENGINE ADBSINX SET DBNO dbno AFD afd
ENGINE ADBSINX SET DBNO dbno AFD afd SEG[NAM] segname
ENGINE ADBSINX SET ?
ENGINE ADBSINX SET DBNO DEFAULT
```

where:

`ADBSINX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`dbno`

Is any valid numeric database value between 0 and 255.

`afd`

Is any valid Access File name.

`segname`

Is any valid ADBS segname in the Access File.

`?`

Queries the current settings.

`DEFAULT`

Returns to the default settings for all previous settings. The DBNO is read from the Access File. If the attributes are not specified in the Access File, the adapter will determine the DBNO from the DDCARD.

Examples for z/OS are shown below:

`ENGINE ADBSINX SET DBNO 5`	Retrieves all data from database number 5.
`ENGINE ADBSINX SET DBNO 2 AFD EMPFILE`	Retrieves data for the EMPFILE from database number 2.

Note: Several commands can be issued for different files. Each command must be issued separately. Changes are cumulative. For example:

```
ENGINE ADBSINX SET DBNO 1 AFD EMPFILE
ENGINE ADBSINX SET DBNO 2 AFD EMP2
ENGINE ADBSINX SET DBNO 3 AFD VEHICLES
```

`ENGINE ADBSINX SET DBNO 2 AFD EMPFILE SEG S02`	Retrieves data for the EMPFILE in database number 2 for a particular ADBS segment (segment S02).
`ENGINE ADBSINX SET ?`	Displays your settings.
`ENGINE ADBSINX SET DBNO DEFAULT`	Returns to the default settings. This resets all SET DBNO commands for all Access Files.

Controlling Data Retrieval Types

The SET CALLTYPE command enables you to set the data retrieval type per session, file, or segment by switching dynamically between physical and logical reads.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Example: Using Dynamic CALLTYPE

The following examples illustrate the use of variations of dynamic CALLTYPE syntax:

```
ENGINE ADBSINX SET CALLTYPE FIND
```

Uses FIND as the type of data retrieval call for all files.

```
ENGINE ADBSINX SET CALLTYPE RL AFD EMPFILE
```

Retrieves data for the EMPFILE using RL as the type of data retrieval call.

Note that you can issue several commands for different files. Each command must be issued separately. Changes are cumulative. For example, the following commands retrieve data for the EMPFILE using FIND as the type of data retrieval call for a particular ADBS segment (segment S02):

```
ENGINE ADBSINX SET CALLTYPE RL AFD EMPFILE

ENGINE ADBSINX SET CALLTYPE FIND AFD EMP2

ENGINE ADBSINX SET CALLTYPE MIXED AFD VEHICLES

ENGINE ADBSINX SET CALLTYPE FIND AFD EMPFILE SEGNAME S02
```

```
ENGINE ADBSINX SET ?
```

Displays your settings.

```
ENGINE ADBSINX SET CALLTYPE DEFAULT
```

Returns to the default settings. This command resets all SET CALLTYPE commands for all Access Files.

LA Fields

The SET LAFORMAT command affects Metadata for fields with LA options:

```
ENGINE ADBSINX SET LAFORMAT {DEFAULT|TX|Annnnn}
```

where:

`DEFAULT`

Maps the ADABAS LA field as fixed-length A500, A500.

`TX`

Maps the ADABAS LA field as text TX50, TX.

A*nnnnn*

(253 < *nnnnn* < 16382) maps LA field with a specified given range length.

UNICODE Fields in W-format

The command sets specified code-page on the field level when mapping ADABAS Unicode W-fields.

```
ENGINE ADBSINX SET WCODEPAGE {DEFAULT|OFF|nnnnn}
```

where:

DEFAULT

Corresponds to codepage 65003.

OFF

Means to process W as A (without conversion).

nnnnn

Means any valid codepage.

Adabas Reporting Considerations

Designers have great flexibility in coding the description of the part of the Adabas structure that they want to access. These topics describe the methods of file traversal that the Adapter for Adabas uses to determine the relative advantages of different file descriptions.

Adabas File Navigation Techniques

When you define specific hierarchical representations of Adabas structures in Master Files, you specify the order in which you want the Adapter for Adabas to retrieve records. This procedure is called navigational logic and is usually part of the application program. Navigation techniques using the JOIN command also work this way.

Referencing Subtrees and Record Retrieval

The Adapter for Adabas selects where to enter the subtree, called the point-of-entry, and the subsequent navigational processing, by analyzing the tree structure defined by your Master File (or JOIN structure) and report request. The adapter determines the smallest subtree that contains all the fields needed for retrieval to produce a report.

The smallest subtree is composed of those segments that contain fields referenced by the request, plus the minimum number of additional segments required to connect all the files used in the request.

The adapter retrieves records only in segments in the referenced subtree. Within the subtree, it retrieves records that contain fields required for the report request or records that are needed to provide the correct links between report fields.

The adapter always enters a database through the root segment of the referenced subtree.

Segment Retrieval Methodology

The Adapter for Adabas retrieves segments from top-to-bottom, then left-to-right at each level of the hierarchy. It retrieves all unique descendant segments before any non-unique descendant segments.

This treatment of unique segments is consistent with a basic server principle: for reporting purposes (though not for updating or file organization), a unique child is logically a direct extension of its parent. This principle is an important factor in selecting a structure that properly reflects your Adabas file. The results of SUM and COUNT operations on fields in child segments may depend on whether they have been declared unique or non-unique. The server also treats missing segments differently, depending on whether the segment is declared unique or non-unique.

Missing Unique Segments

If a segment is specified as unique (SEGTYPE=U or KU), the server regards it as a logical extension of the parent segment. The Adapter for Adabas automatically inserts default values (blanks for alphanumeric fields and zeros for numeric fields) if the unique child segment does not exist. As a result, unique segments are always present.

Syntax: How to Use Missing Non-Unique Segments

If a segment is specified as non-unique (SEGTYPE=S or KM), select one of three options for processing a record without descendant segments

```
SET ALL=all_option
```

where:

all_option
Allows for the processing of records with no descendant segments. Possible values are:

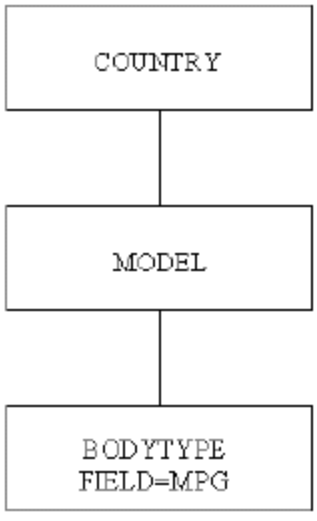
OFF which omits parent instances that are missing descendant segments from the report. OFF is the default value.

ON which includes parent instances that are missing descendant segments in the report.

PASS which includes parent instances that are missing descendant segments, even when IF criteria exist to screen fields in the missing instances of the descendant segment.

You can specify SET ALL in a profile or procedure.

The examples in the following topics describing the SET ALL command are based on the following structure:



SET ALL=OFF

The default option (SET ALL=OFF) rejects a record if the request calls for retrieval of a descendant segment and there is no descendent segment present.

For example, assume you have a file in which the parent segment is COUNTRY, which has a descendant segment named MODEL, which in turn has a descendant segment named BODYTYPE. Using the SET ALL=OFF option, the statement

```
COUNT BODYTYPE BY COUNTRY
```

does not print in the report the details of any country that did not produce at least one bodytype of a model of a car.

SET ALL=ON

The Adapter for Adabas displays the parent record, even if it has no descendant segments. In this case, using the SET ALL=ON option when processing the statement

```
COUNT BODYTYPE BY COUNTRY
```

displays the names of all countries and gives a count of zero (0) bodytypes for those without descendant segments.

However, if the request has an explicit screening test on the descendant segment, the absence of any descendant segments results in test failure. For example, the request

```
COUNT BODYTYPE BY COUNTRY
IF MPG GT 22
```

excludes any country without any bodytype segments from the report.

SET ALL=PASS

The third option, SET ALL=PASS, allows parents without a descendant segment to pass an explicit screening test on that descendant segment. The request

```
COUNT BODYTYPE BY COUNTRY
```

lists all countries with or without bodytype segments. The request

```
COUNT BODYTYPE BY COUNTRY
IF MPG GT 22
```

includes records without any bodytype segments, and those with an MPG greater than or equal to 22.

The ALL Prefix

Selectively apply SET ALL=ON by adding the ALL prefix to any field from the desired segment.

Reference the field either as a sort field (for example, BY ALL.COUNTRY or ACROSS ALL.COUNTRY) or as a display field (WRITE ALL.COUNTRY). As a result, the SET ALL=ON strategy is applied to any missing, immediate, non-unique descendants of the segment containing the ALL-prefixed field. The SET ALL=OFF option is in effect for all other segments.

For example, in the request

```
COUNT MODEL AND BODYTYPE BY ALL.COUNTRY
```

the SET ALL=ON option applies to the country segment and its descendant segments. Therefore, if there is a country without models (and consequently without bodytypes), the report shows that country. Any test condition on either the model or the bodytype segment nullifies the effect of the ALL prefix.

The global SET ALL settings of ON and PASS take precedence over the selective ALL prefix. The selective ALL prefix is effective only when the global setting is OFF, either explicitly or by default.

Adabas Selection Considerations

The Adapter for Adabas analyzes all selection criteria that apply to a specific report request and uses the criteria to minimize its search for data. If a record fails any of the selection tests, the server does not attempt to retrieve any descendant records. Retrieval continues with the next record in the parent segment. If there is no other record in that parent segment and it is not the root of the Master File, The server moves back up to the next record in the previous segment.

The selection tests that you impose on a high-level segment are much more efficient at reducing I/O operations than criteria imposed on lower-level segments. If you know in advance which selection criteria are likely to be used most frequently, design the Master File to take advantage of the hierarchical structure in the Adapter for Adabas.

Selection Order in the Access File

When a report request contains multiple optimizable selection tests, the order of descriptors in the Access File determines the order in which the server applies the selection tests. The server issues a Read Logical (RL) call using the first descriptor listed in the Access File that participates in a selection test.

Therefore, for efficient processing you should describe the most restrictive descriptor at the beginning of its segment in the Access File. The order of descriptors in the Master File has no effect on selection processing.

Syntax: How to Use RECORDLIMIT and READLIMIT

For any request, you may limit the number of Adabas records retrieved from the database and the number of read operations performed. Use the RECORDLIMIT and READLIMIT parameters to impose these limitations by adding the following conditions to a report request

```
{WHERE|IF} RECORDLIMIT {EQ|IS} n {WHERE|IF} READLIMIT {EQ|IS} n
```

where:

n

Is the number of records or read operations at which you want to limit the search.

You add these conditions to any report request, or incorporate them into file security through the DBA facility.

Consider the following example

```
TABLE FILE VEHICLES
PRINT PERSONNEL_ID MAKE MODEL YEAR
WHERE RECORDLIMIT EQ 5
END
```

which produces this report:

```
PAGE      1

PERSONNEL_ID   MAKE            MODEL               YEAR
------------   ----            -----               ----
11500330       CITROEN         BX LEADER             85
11400313       ALFA ROMEO      SPRINT GRAND PRIX     84
30034217       AUSTIN          MINI                  80
30034228       TALBOT          SOLARA                83
30008427       AUSTIN          MINI                  80
```

Notice that only five records are reported as requested.

Syntax: How to Use Optimization of the FIND Call Using Non-Descriptor Fields

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

It is possible to use non-descriptor fields as search criteria and have the calls to Adabas use the search buffer rather than read through the entire database.

The Adapter for Adabas provides improved optimization by allowing the search buffer to be generated using non-descriptor fields. This optimization occurs whenever CALLTYPE=FIND is specified in the Access File and you include an IF or WHERE test referencing a non-descriptor field in your report request.

It may prove to be more efficient to alter your retrieval strategy and perform a Read Physical (L2) call when large amounts of data exist. A SET command is provided for changing the default Adabas call when selecting non-descriptor fields.

```
ENGINE ADBSINX SET NDFIND {ON|OFF}
```

where:

`ENGINE`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Causes the search buffer to be generated with any field (non-descriptor and/or descriptor field). ON is the default value.

`OFF`

Causes the search buffer to be generated with only descriptor fields. If the request does not use any descriptor field, the Read Physical call is generated.

This command applies only if CALLTYPE=FIND is specified in the Access File.

Using the JOIN Command to Process Multiple Files

You can JOIN Adabas databases defined to the server to other Adabas databases or to any other fully joinable database that the server can read. Each JOIN creates a parent-child relationship. The parent field is called the host or *from* field. The child field is called the cross-referenced or *to* field.

You can JOIN to Adabas databases if the cross-referenced field is one of the following:

- A descriptor field.
- A superdescriptor defined with TYPE=SPR or NOP in the Access File.
- A subdescriptor defined with TYPE=NOP in the Access File.

In every case, in the Access File, the cross-referenced segment must specify ACCESS=ADBS.

If CALLTYPE=RL is specified for the cross-referenced segment, the host field can be joined to the high-order portion of a descriptor, superdescriptor, or subdescriptor.

When an Adabas file is the host file, the host field is one of the following:

- A non-recurring field (ACCESS=ADBS).
- A field in a periodic group (ACCESS=PE).
- A multi-value field (ACCESS=MU).

The Adapter for Adabas also supports DEFINE-based JOINs. Up to 16 JOINs can be in effect at one time.

Example: Multi-field JOIN and Short-to-Long JOIN Capability

For a multi-field JOIN, the number of fields used in the JOIN must be the same for both the host and the cross-referenced files. The cross-referenced fields must describe the left-most portion of a superdescriptor defined to the server using the GROUP attribute. Consider the following example.

```
JOIN FLDA AND FLDB IN ADBS1 TO KEY1 AND KEY2 IN ADBS2 AS J1
```

For the short-to-long JOIN, the cross-referenced field must be a descriptor, subdescriptor, or superdescriptor, or a field that describes the left-most portion of a GROUP superdescriptor.

Adabas Optimization With Null-Suppression for CALLTYPE=RL

In the Adapter for Adabas, optimization refers to using an index to retrieve the answer set. To ensure data integrity and complete answer sets, the Adapter for Adabas will perform optimization when all:

- Non-referenced component fields of a superdescriptor are *not* null-suppressed (NU=NO in the Access File).
- Component fields of a superdescriptor that contain null-suppressed fields (NU=YES in the Access File) are used in the selection criteria.

If a field is null-suppressed in Adabas (NU=YES in the Access File), any superdescriptor that uses this field has no entry on the inverted list when this field is blank (alphanumeric) or zero (numeric).

Master File With a Three-Field Superdescriptor

```
GROUP=SUPERG  ,ALIAS=S1 ,USAGE=A9 ,ACTUAL=A9 ,INDEX=I ,$
   FIELD=FLD1 ,ALIAS=AA ,USAGE=A3 ,ACTUAL=A3 ,$
   FIELD=FLD2 ,ALIAS=AB ,USAGE=A3 ,ACTUAL=A3 ,$
   FIELD=FLD3 ,ALIAS=AC ,USAGE=A3 ,ACTUAL=A3 ,$
```

Access File With a Three-Field Superdescriptor

```
FIELD=SUPERG    ,TYPE=SPR,$
   FIELD=FLD1  ,TYPE=   ,NU=YES ,$
   FIELD=FLD2  ,TYPE=   ,NU=NO  ,$
   FIELD=FLD3  ,TYPE=   ,NU=YES ,$
```

In order to optimize a selection test against a superdescriptor with null-suppression, you must explicitly reference the null-suppressed field in the superdescriptor. If you do not reference the null-suppressed field and the field has no data, there is no record in the index and optimization would return no records. To ensure correct results, the server will not optimize the selection test if the null-suppressed field is not referenced.

For more information about null-suppression and how it affects data retrieval, see your Software AG documentation.

Adabas Optimization on Group Fields

If a report request contains IF or WHERE selection tests against one or more fields that describe the left-most portion of a GROUP descriptor, the Adapter for Adabas combines this request into a test that uses the superdescriptor for greater efficiency. If any of the component (or parent) fields of the superdescriptor are defined to Adabas with null-suppression, be sure to note this information in the Access File to ensure accuracy of reads.

For example, consider the following Master File extract:

```
GROUP=SUPERD ,ALIAS=SD ,USAGE=A8 ,ACTUAL=A8 ,INDEX=I ,$
   FIELD=PART1 ,ALIAS=P1 ,USAGE=A4 ,ACTUAL=A4 ,$
   FIELD=PART2 ,ALIAS=P2 ,USAGE=A4 ,ACTUAL=A4 ,$
```

If, in a report request, you include the following two tests,

```
WHERE PART1 EQ 'ABCD'
WHERE PART2 EQ 'EFGH'
```

these two tests are equivalent to the following syntax:

```
WHERE SUPERD EQ 'ABCD/EFGH'
```

This combination uses the superdescriptor's inverted list and optimizes the Adabas call. This optimization is performed only if all null-suppressed (NU=YES) superdescriptor components are explicitly referenced in IF or WHERE tests.

Test on Group Field With Numerics

If you are testing on a group that contains numeric fields, the test must contain the sign byte. The Adapter for Adabas passes only one value per numeric field, based on the preferred sign values in Adabas. A sign value of F is passed for positive numbers and a sign value of D is passed for negative numbers. This sign value reduces the number of calls sent to Adabas and also eliminates the need for Adabas to perform any logical sign translation.

For example:

```
GROUP=GROUP1  ,ALIAS=S1  ,USAGE=A9  ,ACTUAL=A14  ,INDEX=I  ,$
   FIELD=FLD1 ,ALIAS=AA  ,USAGE=A3  ,ACTUAL=A3   ,$
   FIELD=FLD2 ,ALIAS=AB  ,USAGE=P3  ,ACTUAL=P3   ,$
   FIELD=FLD3 ,ALIAS=AC  ,USAGE=A3  ,ACTUAL=A3   ,$
```

In a report request, you must include the sign for the P3 field:

Mainframe platforms:

```
WHERE GROUP1 EQ 'ABC/123F/XYZ'
```

Non-mainframe platforms:

```
WHERE GROUP1 EQ 'ABC/123/XYZ'
```

Adabas Writing Considerations

The Adapter for Adabas is designed to support SQL update commands for an Adabas data source without the use of remote procedures. These topics describe the methods and rules related to write capability.

Adabas Write Adapter

Before you can use any commands that write to an Adabas data source, you must make the following changes to the Master and Access Files:

In the Master File:

- All segments must have SEGTYPE = S0 (for the Read Adapter, SEGTYPE=S is sufficient).
- Fields with ACTUAL format Z (zoned) must have a numeric USAGE format (P or I). Format A is supported only for reading an Adabas data source.

  Note: For the best performance, you can change ACTUAL format Z to ACTUAL format P in the Master File. In this case, you must also change the ALIAS attribute to contain the length and type. If the ALIAS is ff, and the field length is lll, the ALIAS attribute should be coded as:

  ```
  ALIAS='ff,lll,P'
  ```

  For example, consider the following field declaration:

  ```
  FIELD=LEAVE_DUE  ,ALIAS=AU  ,USAGE=P2  ,ACTUAL=P2  ,$
  ```

  You would edit this declarations as follows:

  ```
  FIELD=LEAVE_DUE  ,ALIAS='AU,2,P'  ,USAGE=P2  ,ACTUAL=P2  ,$
  ```

- If two or more fields in the Master File are synonyms (they refer to the same field in the data source and, therefore, have the same ALIAS attribute), only the first field encountered in the Master File can be used in INSERT and UPDATE commands. If a synonym other than the first is used in an INSERT command, it is ignored. If one is used in an UPDATE command, processing is terminated with the following error message:

  ```
  (FOC4565) IGNORED ATTEMPT TO CHANGE NONUPDATABLE FIELD
  ```

In the Access File:

- The segment attribute WRITE=YES indicates that values of fields from the segment may be modified.
- If a segment has the attribute ACCESS=ADBS, you can define a unique key by adding the following attribute:

  ```
  UNQKEYNAME=name
  ```

where:

name

Is the name of the elementary or group field to be used as the unique key.

The UNQKEYNAME attribute does not necessarily define the key described in the ADABAS FDT (the UQ option). The adapter uses it to decide which rules to apply in an INSERT, DELETE, or UPDATE command. If the UNQKEYNAME attribute does not correspond to the key described with the UQ option in the ADABAS FDT, Adabas and the adapter may not agree on whether a segment instance is unique. This can affect the results of INSERT commands. If this attribute is not present, the adapter uses the rules for modifying a segment with a non-unique key or no key. If it is present, the adapter uses the rules for modifying a segment with a unique key. Subsequent sections describe these rules.

- We recommend the use of CALLTYPE=FIND instead of CALLTYPE=RL.

When using the Write Adapter for Adabas, the adapter automatically sets the values of the following two options:

- ADABAS OPEN is set to YES.
- FETCH is set to OFF for all segments.

SQL Commands for the Adapter for Adabas

The Adapter for Adabas supports the following commands:

- SQL INSERT
- SQL UPDATE

  Note: Certain types of fields listed in the Master File cannot be updated. For more information, see *Fields That Cannot be Updated* on page 222.

- SQL DELETE

The adapter issues a COMMIT after each INSERT, UPDATE, or DELETE call. If the Adabas process fails, the adapter issues a ROLLBACK. User rollback of the Adabas process is not supported for SQL commands.

Reference: Obtain Counts of Rows Updated or Deleted

While processing SQL transactions, you can issue the PASSRECS command to obtain counts of rows affected by each successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax: How to Count Records Updated, Inserted or Deleted

```
ENGINE ADBSINX INT SET PASSRECS {ON|OFF}
```

where:

`ADBSINX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`INT`

Indicates that the PASSRECS setting in this command will be applied globally to all adapters that support SQL INSERT, UPDATE, and DELETE commands.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Reference: Fields That Cannot be Updated

The SQL UPDATE command for the following types of fields is ignored:

- Counter fields (ALIAS=xxC).
- ORDER fields (ALIAS=ORDER).
- Group fields.
- Unique key fields (fields specified by the UNQKEYNAME attributes in the Access File).

The SQL UPDATE command for the following types of fields produces an error:

- Synonym fields.
- Fields created from subdescriptors or superdescriptors (as defined in the Access File).

Reference: **Using Synonym Fields (Not Related to Creating a Synonym)**

In a Master File, two or more field declarations can refer to the same Adabas field. Each duplicate field declaration after the first is called a synonym field. Such synonym fields can be used in report commands, but cannot be used in write commands. The following actions occur as a result of using synonyms in write commands:

- Synonym fields used in SQL INSERT are ignored.
- Synonym fields used in SQL UPDATE cause processing to terminate and generate the message:

  ```
  (FOC4565) IGNORED ATTEMPT TO CHANGE NONUPDATABLE FIELD
  ```

Syntax: **How to Insert Records Into an Adabas Data Source**

```
INSERT INTO mfname [(field1, field2, ...)]
VALUES ('value1','value2', ...)
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Is an optional list of fields to be inserted. If you omit the list, the value list must contain values for all fields in all segments in the Master File as long as the Master File only specifies a single path Adabas file. If the Master File is multi-path, the field list is required and can specify only a single path. Elementary fields not included in the field list have empty values in the data source. The field list can contain only elementary fields. Groups, subdescriptors, superdescriptors, and hyperdescriptors, cannot be used in the field list.

'value1', 'value2',...

Is the list of values to be inserted. Each alphanumeric value must be enclosed in single quotation marks. If you specify the field name list, the value list only needs to specify, in field name list order, the values for the field names supplied. At a minimum, you must supply all the key fields for all the segments in the path to the target.

If a segment is not present, default values are generated for all fields that are not supplied, and the missing path segments are inserted along with the target segment.

Reference: Rules for Inserting Records Into an Adabas Data Source

- For a segment with a unique key:

  The key field value is used to insert the target segment. If any additional fieldname=value pairs are supplied for a segment in the path, they are used to qualify that path segment. If a segment with ACCESS=ADBS has a unique key or group of keys, only these key fields should be used in the INSERT command. All other fields should be added to the record using the UPDATE command.

- For a segment with a non-unique key or no key:

  If you want to insert an additional record for an existing key field, you must have at least one field in addition to the key field in the field list in order to make the record unique. If you do not, the insert is rejected.

- For segments with ACCESS=PE or MU, if a new occurrence is inserted, you must set the occurrence number (ORDER field) to 0 (zero indicates the next occurrence) or to a value greater than the number of existing occurrences. This new occurrence is always inserted after the last existing occurrence. For example, if a PE or MU segment has two existing occurrences, the next occurrence added is always the third. If the occurrence with the given number already exists, processing terminates with the following message:

  ```
  (FOC4564) THIS OCCURRENCE ALREADY EXISTS. USE UPDATE COMMAND
  ```

  You should use the UPDATE command instead of INSERT in this case.

- For segments in which the parent segment contains ACCESS=PE, and the child segment contains ACCESS=MU without the NU option in the ADABAS FDT:

  - If an INSERT command is issued for the parent (PE) segment alone, Adabas automatically inserts an empty child (MU) segment. You can use the UPDATE command to provide values for this child.

  - If one INSERT command is issued for the parent (PE) and child (MU) segments simultaneously, the first occurrence in the child segment is inserted by the Write Adapter for Adabas using the values from the INSERT command.

- For segments in which the parent segment has ACCESS=PE, and the child segment has ACCESS=MU with the NU option in the ADABAS FDT, the empty child is automatically suppressed by Adabas. You can use an INSERT command for the parent and child segments separately or simultaneously.

Reference: Effect of UNQKEYNAME on INSERT Actions

The UNQKEYNAME attribute in the Access File determines how the adapter presents an INSERT command to Adabas. The UQ option in the ADABAS FDT and the specific field values listed in the INSERT command determine whether Adabas actually inserts the segment instance. The following table describes how these factors affect the result of the INSERT command. Assume that the Access File specifies UNQKEYNAME=EMPLOYEE_ID and that the employee ID value EMPID005 already exists in the data source:

Result of the INSERT Command for Existing EMPLOYEE_ID EDMPID005

EMPLOYEE_ID has the UQ Option in FDT	Fields in INSERT Command	Instance Inserted
No	EMPLOYEE_ID only.	No - rejected duplicate
Yes	EMPLOYEE_ID only.	No - rejected duplicate
No	EMPLOYEE_ID plus fields with values that do not already exist.	Yes
Yes	EMPLOYEE_ID plus fields with values that do not already exist.	No - message (FOC4561), RC=198

Result of INSERT Command when EMPLOYEE_ID is not in the field list

UNQKEYNAME = EMPLOYEE_ID	Instance Inserted
Yes	No - message (FOC4563)
No	Yes, with empty EMPLOYEE_ID value.

Syntax: How to Update Field Values in an Adabas Data Source

```
UPDATE mfname SET field1 ='value1' [,field2 ='value2'...]
WHERE kfield1 ='kvalue1' [AND kfield2 ='kvalue2'...]
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Are the names of the fields to be updated.

'value1', 'value2',...

Are the new values for the updated fields, enclosed in single quotation marks.

kfield1, kfield2, ...

Are, at a minimum, all of the key fields for all the segments in the path to the target. Only one target segment is updated. You can include additional fieldname=value pairs for the target or any segment in the path. When additional fieldname=value pairs are present for the target segment, these are used for change verification protocol processing. If all target fieldname=value pairs that are present match the Adabas segment field values, the segment is updated with the SET fieldname values. The field list can contain only elementary fields. Groups, subdescriptors, superdescriptors, and hyperdescriptors, cannot be used in the field list.

'kvalue1', 'kvalue2',...

Are the values that identify the target segment, enclosed in single quotation marks.

Reference: Rules for Updating Records in an Adabas Data Source

- For a segment with a unique key, the key field value and any additional fieldname=value pairs are used to qualify the target segment for an update.
- For a segment with a non-unique key, you must supply the key field. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is updated. The use of Adabas descriptors for this type of segment is highly recommended for efficiency.
- For a segment with no key, you must supply at least one fieldname=value pair. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is updated.

Syntax: How to Delete Records From an Adabas Data Source

```
DELETE FROM mfname WHERE field1 ='value1' [AND field2 ='value2'...]
```

where:

mfname

Is the name of the Master File to use.

field1, field2, ...

Are, at a minimum, the names of all of the key fields for all the segments in the path to the target. Only one target segment is deleted; Adabas cascades the delete to dependent segments. Additional fieldname=value pairs may be included for the target segment and any segment in the path that has no key or a non-unique key. It is recommended to use Adabas descriptor fields for any additional fieldname=value pairs.

'value1', 'value2',...

Are the values that identify the target segment to be deleted.

Reference: **Rules for Deleting Records From an Adabas Data Source**

- For a segment with a unique key, the key field value is used to delete the target segment.
- For a segment with a non-unique key, you must supply the key field. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is deleted. The use of Adabas descriptors for this type of segment is highly recommended for efficiency.
- For a segment with no key, you must supply at least one fieldname=value pair. If the WHERE criteria for this type of segment, in the path or target, do not identify a unique occurrence, the first occurrence found is deleted.
- A segment occurrence with ACCESS=PE or MU is deleted from the data source except if it is the last occurrence for an ACCESS=PE segment. Adabas only deletes the last occurrence if all fields have the NU option in the FDT. If they do not all have this option, the occurrence has empty values in all fields.
- When you delete segments that have dependent segments, the DELETED counter for the session may have an incorrect value. For a first level segment with descendants, this counter is always incorrect. For a second level segment, this counter is incorrect if there are multiple descendant segments. For a third level segment, this counter is always correct.

Adapter Navigation

To retrieve the necessary records that fulfill a request, the Adapter for Adabas navigates the Adabas database through direct calls in read-only mode. The calls generated depend on three factors:

- Information supplied in the Master File.
- Information supplied in the Access File for a specified segment.

- Particulars of the report request.

The adapter uses information from these sources to choose the most appropriate and efficient retrieval method.

There are three logical types of Adabas access:

- Entry segment retrieval of Adabas records.
- Retrieval of descendant periodic groups and multi-value fields.
- Retrieval of descendant Adabas records.

Entry Segment Retrieval of Adabas Records

The server constructs an Adabas request based on the Master File, Access File, and the report request. The root of the subtree is the entry segment into the database for a particular request.

For Adabas structures, the root of the subtree must be a segment in the Access File containing singly occurring fields (ACCESS=ADBS). The Adabas calls generated to retrieve data for this entry segment depend on the Access File information that has been supplied for the segment and the selection tests in the request.

The first decision the Adapter for Adabas makes for the entry segment is whether to retrieve all the records of the segment or just a subset.

All records are retrieved for a segment if either of the following conditions exists:

- There is *no* selection test on any field that is a descriptor (defined in the Access File with TYPE=SPR, DSC, or NOP) and the SEQFIELD has not been defined.
- The specified selection test is *not* IF or WHERE, and does not use the relational operators:

  `IS, EQ, GE, GT, LT, LE, or FROM...TO`

  For example:

  field **IS** *value1* [(OR *value2*... OR *valuen*)]

  field **IS** (*ddname*)

  field **FROM** *value1* **TO** *value2* [(OR *value3* TO *value4*...)]

  field **GE** *value1*

  field **GT** *value1*

```
field LT value1

field LE value1

group EQ 'value1/value2/value3'
```

When you are using one or more of these selection tests against a descriptor, the adapter uses the inverted list to retrieve a subset of the records for the segment.

For example, consider the following selection criterion:

```
field IS value1 [(OR value2... OR valuen)]
```

In this example, any combination of full values or partial values triggers the use of inverted lists except when the $ (used to indicate a masked field) is in the first position of the value. If the $ is in the first position of the value, the search conducted by Adabas is not done through the descriptor's inverted list. Instead, the search is conducted physically through the database in the same manner that a Read Physical call performs a retrieval. See *Read Physical Calls* on page 230 for more information.

Once the adapter determines whether to retrieve all or a subset of the segment records, it decides which access strategy to use based on the Access File attribute settings. The following are the strategies that the adapter can use:

- Retrieval of all records through Read Physical (L2) calls discussed in *Read Physical Calls* on page 230.
- Retrieval of all records through Read Logical (L3) calls without using a starting value discussed in *Read Logical Navigation* on page 231.
- Retrieval of a subset of records through Read Logical (L3) calls using a starting value discussed in *Read Logical Navigation* on page 231.
- Retrieval of all records for the entry segment through Read Logical by ISN (L1) calls discussed in *Read Logical Navigation* on page 231.
- Retrieval of a subset of records through simple FIND (S1) calls discussed in *FIND Navigation* on page 235.
- Retrieval of a subset of records through complex FIND (S1) calls discussed in *FIND Navigation* on page 235.
- Retrieval of all records through Read Descriptor Value (L9) direct calls discussed in *Read Descriptor Value (L9) Direct Calls* on page 238.

The selection logic is designed to implement as many of the record selection tests as possible at the Adabas level. It minimizes the number of I/O operations required to access the necessary data by issuing efficient calls to Adabas.

Read Physical Calls

Read Physical calls read every database record in a physical file, and then return every record for the entry segment to the server. The server must then select the records that meet the selection criteria in the request and discard the rest.

The Adapter for Adabas issues Read Physical (L2) calls to retrieve all records for the entry segment when:

- The request contains no optimizable selection tests on an inverted list (descriptor) and the Access File does not have a SEQFIELD defined.
- With CALLTYPE=FIND, there is no screening on a descriptor, the screening for a non-descriptor has been set off, and the Access File does not have a SEQFIELD defined.

See *Adabas Reporting Considerations* on page 211 for more information about retrieval of records with no screening on a descriptor.

The steps the adapter and Adabas perform to complete a Read Physical call are:

1. The adapter constructs a Read Physical (L2) call.
2. Adabas returns each record in the physical file corresponding to the Adabas file number specified. The records are retrieved in the order in which they are physically stored.

Subsequent retrieval for the entry segment is completed by issuing the same Read Physical call with the User Control Block unmodified. The adapter terminates retrieval when one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached (see *Adabas Reporting Considerations* on page 211 for information about these keywords).

Read Physical (L2) calls can be faster than Read Logical (L3) calls depending on the request and the data dispersion of the physical storage. Ask your Adabas database administrator if you should use SEQFIELD for a given Adabas record, or if Read Physical would be more efficient.

Read Logical Navigation

When designing a database, it is necessary to know the contents of the files being created. Defining descriptors is very important for efficiency. The Adapter for Adabas does not have information about the different fields, or about which field is more efficient to parse first, unless the Access File has this information. The Access File tells the adapter how to access your data in the most efficient manner. The order of the fields (for example, DSC, NOP, or SPR descriptors) is important in the Access File, as the adapter uses this order when executing an IF or WHERE clause.

The Access File needs to have information about the index fields used in the report request. The order in which you define these fields is important. Adabas is field oriented, not positional. The order of the fields will not affect retrieval, except for the selection of which inverted list to use.

The Access File controls the order of the IF or WHERE clauses, so the database administrator can set up the Access File in the order of the most efficient reads. The user will then receive the requested data much more efficiently. The order in the Access File controls the order of retrieval.

Read Logical Calls Without a Starting Value

The Adapter for Adabas uses Read Logical (L3) calls to retrieve all records for the entry segment when the Access File contains a SEQFIELD value for that segment and the report request does not contain an optimizable selection test on an inverted list, and CALLTYPE=RL.

SEQFIELD is generally used to suppress Read Physical calls when there is no optimizable selection test on an inverted list. A Read Physical call will be issued if a SEQFIELD is not defined.

The steps the adapter and Adabas perform to complete a Read Logical call without starting values are:

1. The adapter constructs a Read Logical (L3) call without the 'Value Start' option and without the Adabas Search and Value buffers.
2. Adabas returns each record corresponding to an entry in the inverted list. The records are in the same order as the inverted list.

Subsequent retrieval for the entry segment is done by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list. The adapter terminates retrieval when one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

In an Adabas file that has no record types, you can select the value of SEQFIELD from any inverted list containing entries for all records on the file.

SEQFIELD is required when the Adabas file has several record types. In this case, the value of SEQFIELD is an inverted list which has entries for a single record type.

If the descriptor used as the SEQFIELD is null-suppressed or contains null-suppressed fields, only records where the descriptor is populated will be included in the retrieval. An entry is not included in the inverted list if a field is null and will not be returned by Adabas to the server. See your Software AG documentation for more information about null-suppression and how it affects data retrieval.

Read Logical Calls With a Starting Value

The Adapter for Adabas constructs Read Logical (L3) calls for the root of the accessed subtree when the following conditions exist:

- The Access File contains CALLTYPE=RL for that segment.
- The request contains an optimizable selection test against a descriptor or superdescriptor identified by TYPE=SPR or DSC in the Access File.
- None of the selection tests is on a field identified by TYPE=NOP in the Access File.

When a report request contains multiple optimizable selection tests, the order of descriptors in the Access File determines the order in which the server applies the selection tests. The server issues a Read Logical (RL) call using the first descriptor listed in the Access File that participates in a selection test.

Therefore, for efficient processing, you should describe the most restrictive descriptor at the beginning of its segment in the Access File. The order of descriptors in the Master File has no effect on selection processing.

The steps Adabas performs to complete a Read Logical call are:

1. The adapter constructs a Read Logical (L3) call with the starting value to be retrieved for that inverted list.
2. The adapter calls Adabas with a 'V' in Command Option 2, which instructs Adabas to use the 'Value Start' option.
3. The call retrieves the first record whose value in the inverted list is equal to or greater than the value specified in the request.

Subsequent retrieval for the entry segment is done by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list.

The adapter terminates retrieval when one of the following occurs:

- The value of the inverted list is out of range of the request.
- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

If there are multiple values specified in the request (for example, WHERE *field* EQ *value* OR *value*..., WHERE *field* FROM *val1* TO *val2* OR *val3* TO *val4*) and RECORDLIMIT or READLIMIT has not been reached, the Adapter for Adabas releases the Command ID and reissues the Read Logical call with the next starting value. For each set of values, the adapter terminates retrieval when one of the above conditions occurs.

Retrieval Through Read Logical by ISN (L1) Calls

The Adapter for Adabas uses Read Logical by ISN (L1) calls to retrieve all records for the entry segment when the:

- Report request does not contain an optimizable selection test on an inverted list or an ISN list.
- Access File contains a SEQFIELD value for that segment and this value is equal to the ISN field name.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN. For example:

Access File ADATEST

```
SEGNAM=S01, ACCESS=ADBS, ... , SEQFIELD=ISN_FIELD ,$
```

Request

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST;
```

The adapter uses Read Logical by ISN (L1) calls to retrieve a single record for the entry segment when the:

- Report request contains the only EQ relational operator in the selection test on an ISN list.
- ISN field is used as the cross-referenced field in the Join to Adabas file.

Adabas returns RC 113 if the record with the ISN defined in the test is not present in the Address Converter for the file. Then the adapter returns the answer "Record is not found".

For example:

```
SELECT AA_FIELD, AJ_FIELD FROM ADATEST
WHERE ISN_FIELD = 1100;
```

Note: The Multifetch option will be suppressed for this call.

The adapter uses Read Logical by ISN (L1) calls to retrieve subset of records for the entry segment when the:

- Report request contains the only GE/GT relational operator in the selection test on an ISN list.
- Report request does not contain any selection test on an inverted list.
- Access File contains a SEQFIELD value for that segment.

Adabas returns each record corresponding to an entry in the Address Converter. The records are in ascending value of the ISN.

For example, the Access File ADATEST contains:

```
SEGNAM=S01, ACCESS=ADBS , ... ,SEQFIELD=AA_FIELD ,$
```

Request 1. When Read Logical by ISN (L1) used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100;
```

Request 2. When Read Logical (L3) used:

```
SELECT AA_FIELD, AE_FIELD FROM ADATEST
WHERE ISN_FIELD > 1100 AND AJ_FIELD = 'TAMPA';
```

After issuing an Insert request to the file with the ISN field in the Master File, the resulting ISN from Adabas is printed in the message FOC4592:

```
(FOC4592) RECORD IS INSERTED WITH ISN : nnnnn
```

For example:

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD)
VALUES ('11111111', 'TAMPA');
END


ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 11
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

If the Insert request contains an ISN field in the fields list and a corresponding value is not 0, then the adapter issues an N2 call to Adabas. Adabas assigns this ISN value to the record provided by the user. Adabas returns RC 113 if this value was already assigned to another record in the file or if it is larger than the MAXISN in effect for the file.

For example:

Request 1

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD, ISN_FIELD)
VALUES ('11111114', 'TAMPA', 14);
END

ADBTEST ADBSINX ON 09/20/2002 AT 15.22.16
(FOC4592) RECORD IS INSERTED WITH ISN : 14
(FOC4566) RECORDS AFFECTED DURING CURRENT REQUEST : 1/INSERT
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 1 UPDATED = 0 DELETED = 0
```

Request 2

```
SQL
INSERT INTO ADBTEST (AA_FIELD, AJ_FIELD, ISN_FIELD)
VALUES ('11111114', 'TAMPA', 999999);
END

ADBEMP43ADBSIN ON 09/19/2002 AT 16.16.22
(FOC4561) ERROR IN ADABAS INCLUDE /113
TRANSACTIONS: TOTAL = 1 ACCEPTED= 1 REJECTED= 0
SEGMENTS: INPUT = 0 UPDATED = 0 DELETED = 0
```

FIND Navigation

The manipulation of ISN lists is done on the initial call to Adabas. Both the intermediate lists and the resulting list may be very large and may take up a large percentage of the Adabas workspace. The Adabas work area is shared by all Adabas users, and complex FINDs may affect programs that are updating the database. You may want to suppress FINDs altogether. Accomplish this suppression by specifying CALLTYPE=RL on every Access File segment. Ask your Adabas database administrator whether to use FIND processing.

Simple FIND Calls

The Adapter for Adabas constructs a simple FIND (S1) call, using a single inverted list structure, for the root of the accessed subtree if one of the following conditions is met:

- The Access File contains either CALLTYPE=FIND for that segment or does not specify the CALLTYPE attribute, in which case the adapter default is CALLTYPE=FIND. Additionally, the request contains a single selection test on a descriptor identified with TYPE=DSC or a superdescriptor identified with TYPE=SPR in the Access File.
- For any value of CALLTYPE (RL or FIND), if the request contains a single selection test on a subdescriptor or superdescriptor containing partial fields (identified with TYPE=NOP in the Access File).

CALLTYPE=FIND instructs the adapter to generate FIND calls to retrieve records from the inverted lists. This method is the default for processing selection criteria on inverted lists for a segment (if CALLTYPE is omitted from the Access File segment declaration).

If you have selected a field defined with TYPE=NOP (for example, subdescriptors), a FIND call is issued even if RL has been specified in cases when:

- SET NOPACCESS FIND was performed.
- The server is running in an OpenVMS environment.
- The same Adabas field is defined in a descendant PE/MU segment.

Switching from RL to FIND mode is also performed if a:

- Field is defined with TYPE=SPR and the server is running in an OpenVMS environment.
- Field is defined with TYPE=PDS (phonetic descriptor) or TYPE=HDS (hyperdescriptor).
- Field is defined in a PE segment (that it is a part of periodic group).

For performance considerations, a FIND (S1) call does not issue a READ ISN (L1) call if an answer set containing only one record is returned. The GET FIRST option returns the first record in the inverted list. This reduces unnecessary I/O calls. If the FIND call returns more than one ISN in a list, the GET NEXT option of L1 is used to start reading the ISN list from the second entry.

The steps Adabas performs to complete a FIND (S1) call are:

1. The adapter constructs a FIND (S1) call with the value(s) specified.
2. The adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of Internal Sequence Numbers (ISNs) in the Adabas work area.

3. Adabas constructs a complete list of ISNs for every record matching the selection criteria in the work area of Adabas. This list is sorted in ascending ISN order.
4. The adapter issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria. The records are returned in ascending ISN order.

All subsequent retrieval for this entry segment is done using the Read ISN (L1) call issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Complex FIND Calls

The Adapter for Adabas constructs complex FIND (S1) calls, using two or more inverted list structures, for the root of the accessed subtree if one of the following conditions is met:

- The Access File contains CALLTYPE=FIND for that segment (or omits CALLTYPE). The request contains multiple selection tests on a descriptor or superdescriptor described with TYPE=DSC or TYPE=SPR in the Access File.
- For any value of CALLTYPE, the request contains selection tests on a subdescriptor or superdescriptor containing partial fields (identified with TYPE=NOP in the Access File).

The steps Adabas performs to complete a FIND call on multiple inverted list structures are:

1. The adapter constructs a FIND (S1) call with all the values specified for each inverted list on which there are selection criteria.
2. The adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
3. Adabas constructs an ISN list for each inverted list specified.
4. Adabas merges these lists into a final list of ISNs which match all of the selection criteria in the call. This list is kept in the Adabas work area and is sorted in ascending ISN order.
5. The adapter then issues a Read ISN (L1) call to retrieve the first record from the list in the Adabas work area. The records are returned in ascending ISN order.

All subsequent retrieval for this entry segment is completed using Read ISN (L1) calls issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Read Descriptor Value (L9) Direct Calls

The Adapter for Adabas issues the Adabas Read Descriptor Value (L9) direct call when you use the COUNT command or SUM CNT.*field* within a report request. L9 calls are, by definition, limited to descriptors defined to the server with the following:

- INDEX=I in the Master File.
- A field type of NOP, DSC, or SPR (defined in the Access File).

L9 calls allow an aggregate ISN count to be returned for each unique value on the inverted list at a cost of only one call per unique value. This technique allows a dramatic efficiency gain over passing each record to the server to process the count.

Selection criteria (for example, WHERE *descriptor* EQ '1234') are passed along with the L9 call to further limit the number of calls. If any non-descriptor fields are mentioned in the TABLE request, a Read Descriptor Value (L9) direct call is not possible and the server uses its own internal count processing. If a BY field is used, it must be the same field or GROUP name used as the object of the COUNT verb.

Here is an example of the SUM CNT.*field* using EMPFILE1:

```
TABLE FILE EMPFILE1
SUM CNT.DEPT_S03
BY DEPT_S03
END
```

Descendant Periodic Groups and Multi-Value Fields

The Adapter for Adabas must retrieve a count of the number of occurrences before attempting to retrieve a periodic group (PE) or multi-value (MU) field. This retrieval is accomplished by taking the appropriate Adabas counter field (indicated by the OCCURS attribute on the segment declaration in the Master File) and placing its ALIAS (the two-character Adabas name of the PE or MU appended with the letter C) in the Format buffer of the call for the parent record fields.

Adabas returns to this counter field the total number of occurrences that exist for the PE or MU.

The adapter retrieves the descendant data in one of the following ways:

- For MU fields or PE groups which do not contain an MU, all the occurrences are retrieved at once. The counter field ALIAS is used without the letter C, to retrieve the entire set of occurrences in one Read ISN (L2) call.

- For PE groups which contain one or more MU fields, the adapter retrieves a single occurrence of all component fields at a time, including the count(s) of the number of MU descendants, if required. For example, a PE group containing an MU field which has five occurrences of the PE requires five Read ISN (L2) calls to retrieve all of them.

Descendant Adabas Records

The access strategy used to retrieve descendant records in a subtree depends on the Access File attributes specified for a given segment and the SEGTYPE attribute in the Master File (or the SEGTYPE created on the cross-referenced segment as the result of a JOIN). In some cases, the selection criteria in a request further affect access strategy for descendant segments.

Descendant records are related to their parents by a field or GROUP in the parent that corresponds to a field or GROUP in the descendant. This relationship is set up either in the Access File using the KEYFLD/IXFLD pair or through the host field/cross-referenced field combination in a JOIN.

The most appropriate access strategy is selected based on the value of CALLTYPE of the descendent segment in the Access File. The three basic strategies used to obtain descendant records are:

- Retrieval of descendant records through Read Logical calls using a starting value.
- Retrieval of descendant records through simple FIND calls.
- Retrieval of descendant records through complex FIND calls.

The CALLTYPE attribute, the KEYFLD/IXFLD attributes, the SEGTYPE, and the TABLE request are the determining factors in the way descendant data is retrieved.

Read Logical Calls Using a Starting Value

The Adapter for Adabas constructs Read Logical (L3) calls for related descendant records of a parent when the following conditions are met:

- The Access File contains CALLTYPE=RL for that segment.
- The IXFLD or JOIN *to* field is described as TYPE=DSC or TYPE=SPR in the Access File.
- No other selection criteria on inverted lists for this segment have selection on a descriptor with TYPE=NOP.

Read Logical processing is performed on a single inverted list only. The steps the adapter performs to complete a Read Logical call using a starting value are:

1. The adapter constructs a Read Logical (L3) call with the value of the KEYFLD from the parent segment for the IXFLD inverted list. This call retrieves the first record whose value in the inverted list is equal to or greater than the value of the KEYFLD.
2. For unique descendants, no additional retrieval is performed on the descendant segment for any given parent. (For an embedded cross-referenced segment, the SEGTYPE is U or KU or the segment was the cross-referenced segment in a unique JOIN.)
3. For non-unique descendants, subsequent retrieval for the entry segment is completed by issuing the same Read Logical call with the User Control Block unmodified. The records are returned in ascending value of the inverted list.

The adapter terminates retrieval when one of the following conditions is met:

- The value of the inverted list is greater than the value of the KEYFLD.
- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Simple FIND Calls (Descendant Records)

The Adapter for Adabas constructs a simple FIND (S1) call, using a single inverted list, to obtain related descendant records for a parent when one of the following conditions exists:

- The Access File contains CALLTYPE=FIND (or omits CALLTYPE) and the request contains no selection criteria on the descendant segment.
- The Access File contains CALLTYPE=FIND (or omits CALLTYPE) and the request contains selection criteria on the descendant segment and the descendant is unique. (For an embedded cross-referenced segment, the SEGTYPE is U or KU or the segment was the cross-referenced segment in a unique JOIN.)
- For any value of CALLTYPE, IXFLD is described with TYPE=NOP in the Access File and the request contains no selection criteria on the descendant segment.

The steps Adabas performs to complete simple FIND calls against descendant segments are:

1. The adapter constructs a FIND (S1) call with the value of the KEYFLD from the parent segment for the inverted IXFLD list.
2. For unique descendants, Adabas returns the record on the FIND call; no 'H' command is issued to Adabas. Only one call will be issued, and only the first instance of data will be returned.
3. For non-unique descendants, the adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.

4. Adabas constructs a complete list of ISNs for every record related to the parent record in the work area. This list is sorted in ascending ISN order.
5. The adapter issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria.

If the descendant is not unique, subsequent retrieval of records for this descendant segment is completed using the Read ISN (L1) call issued against the ISN list held by Adabas. L1 calls are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Complex FIND Calls (Descendant Records)

The Adapter for Adabas constructs a complex FIND (S1) call, using several inverted list structures, to obtain related descendant records for a parent when one of the following conditions is met:

- The Access File contains CALLTYPE=FIND for that segment (or omits CALLTYPE) and the descendant is non-unique. (For an embedded cross-referenced segment, the SEGTYPE is S or KM, or the segment was the cross-referenced segment in a non-unique JOIN.) The report request contains one or more selection criteria on the descendant segment.
- The Access File contains any value of CALLTYPE. The descendant is non-unique. (For an embedded cross-referenced segment, the SEGTYPE is S or KM, or the segment was the cross-referenced segment in a non-unique JOIN.) The request contains one or more selection criteria on the descendant segment, one of which is on a descriptor, or the IXFLD is described as TYPE=NOP in the Access File.

The steps Adabas performs to complete a complex FIND call are:

1. The adapter constructs a FIND (S1) call with the value of the KEYFLD from the parent segment using the inverted IXFLD list, in addition to all other selection criteria on inverted lists from the request.
2. For non-unique descendants, the adapter calls Adabas with an 'H' in Command Option 1, which instructs Adabas to store the resulting list of ISNs in the Adabas work area.
3. Adabas constructs an ISN list for each inverted list specified, and merges these lists into a final list of ISNs that match all of the selection criteria in the call. This list is kept in the Adabas work area and is sorted in ascending ISN order.
4. The adapter then issues a Read ISN (L1) call to retrieve the first record from the Adabas work area which matches the selection criteria.

Subsequent retrieval of records for this descendant segment is done through the Read ISN (L1) call issued against the ISN list held by Adabas. L1 commands are issued until one of the following occurs:

- Adabas issues a response code indicating end-of-list.
- The RECORDLIMIT or READLIMIT is reached.

Chapter 4

Using the Adapter for Adabas Stored Procedures

The Adapter for Adabas Stored Procedures enables you to invoke an Adabas stored procedure, which is a Natural subprogram created and tested using Natural facilities.

Note: This adapter is only available on the z/OS operating systems.

You invoke a stored procedure by:

- Issuing a data retrieval command in an SQL request.
- Issuing a Data Manipulation Language (DML) request. (DML is Information Builders internal retrieval language, TABLE.)
- Executing a remote procedure call.

This enables you to invoke an Adabas stored procedure from environments like WebFOCUS and DataMigrator. You can pass input parameters to a stored procedure by specifying filtering criteria in the request or to an RPC by passing a parameter list. You retrieve output as an answer set.

You use the Web Console or the Data Management Console to generate a synonym for a stored procedure using information from Natural data areas. The synonym maps the input and output parameters of a procedure.

In this chapter:

Preparing the Adabas Stored Procedures Environment

The Adapter for Adabas must be installed and configured prior to configuring the Adapter for Adabas Stored Procedures. For information about configuring the Adapter for Adabas, see *Using the Adapter for Adabas* on page 129.

Configuring the Adapter for Adabas Stored Procedures

You can configure the Adapter for Adabas Stored Procedures from the Web Console or the Data Management Console.

Procedure: How to Configure the Adapter

You must have configured the Adapter for Adabas in order to be able to configure the Adapter for Adabas Stored Procedures. For configuration instructions, see *Using the Adapter for Adabas* on page 129.

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Note: The Adapter for Adabas Stored Procedures is under the *Procedures* group folder. No input parameters are required.

Managing Adabas Stored Procedure Metadata

When the adapter invokes an Adabas stored procedure, it needs to know where to find the stored procedure and how to process its parameters. For each stored procedure the adapter will access, you create a synonym that describes this information.

Reference: Requirements for Creating a Synonym

Creating a synonym, that is, metadata, for Adabas Stored Procedures consists of mapping input and output parameters that are described in a Natural library as local or global (LDA/GDA) data areas.

Only the first record of a definition can be level 1. If a data area has more than one parameter, the parameters must be defined as a structure.

Example: Defining One Parameter in an Adabas Stored Procedure

```
Local     SAMPPRM1 Library SYSRPC                DBID     3 FNR     8
Command                                                            > +
I T L Name                           F Leng Index/Init/EM/Name/Comment
All - ------------------------------ - ---- -------------------------
    1 PARAMETER                      A   10

------------------------------------------------------------ S 1    L 1
```

Example: Defining Multiple Parameters in an Adabas Stored Procedure

```
Local     SAMPPRM2 Library SYSRPC                DBID     3 FNR     8
Command                                                            > +
I T L Name                           F Leng Index/Init/EM/Name/Comment
All - ------------------------------ - ---- -------------------------
    1 PARAMETERS
    2 TEXTINP                        A   60
    2 TEXTOUT                        A   60
    2 KEYWORD                        A   20 (1:20)

------------------------------------------------------------ S 4    L 1
```

Creating Synonyms

A synonym defines a unique logical name (also known as an alias) for each Adabas stored procedure, and one set of input/output parameters. The adapter requires that you generate a synonym for each Adabas stored procedure that you want to invoke with the adapter.

Synonyms are also useful because they insulate client applications from changes to the location of a procedure. You can move or rename a procedure without modifying the client applications that use it. You need to make only one change, redefining the synonym used by the procedure on the server.

Creating a synonym generates a Master File and an Access File: these are metadata that describe the name, parameters, and options of the Adabas stored procedure to the server.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Adabas Stored Procedure

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Data Base ID

The database ID.

Natural System System File

A number that identifies the Natural System system file.

Natural Stored Procedure Library

Natural library with stored procedure.

Enter a string for filtering library names, inserting the wildcard character (%) at the beginning and/or end of the string.

For example, enter ABC% to display libraries whose names begin with the letters ABC; %ABC to display libraries whose names end with the letters ABC;%ABC% to display libraries whose names contain the letters ABC at the beginning, middle, or end; or % to display all libraries.

Mask for Subprogram Name

The mask for returning a list of subprograms from which to chose. Enter a string for filtering subprogram names, inserting the wildcard character (%) at the beginning and/or end of the string.

Subprogram names

Choose a program from the drop-down list.

To identify the request buffer for the stored procedure, enter values for the next three parameters:

Data Base ID

The database ID.

Natural System User File

A number that identifies the Natural System user file.

Mask for the Libraries

The mask for returning a list of libraries from which to chose the one containing local/global data areas that describe the parameters of the stored procedure.

Mask for the Data area names

Enter the mask for returning a list of the data areas from which you will chose the data area defining the stored procedure parameters.

Natural Libraries

Select a library from the drop-down list.

Trigger File Data Base ID

The trigger file database ID.

Trigger File Number

The trigger file number.

Transaction Option

Specifies the stored procedure transaction property.

Parameter Option

Specifies the type of parameters the stored procedure has: no parameters, control, or error message.

Synonym name

The name of the synonym. You can retain the current name or change it.

Changing a synonym name enables you to manage multiple synonym versions to reflect, for example, multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Providing a suffix enables you to manage multiple synonym versions to reflect multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Input Data Area/ Output Data Area

Specifies which data areas are for input parameters and which are for output parameters. You can specify the same data area as the source of both the input and output parameters.

You can specify a data area only if Parameter Option is set to a value other than None.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Example: Generating a Synonym

The following sample Master File and Access File are generated from the Web Console using the Create Synonym facility.

Generated Master File SAMP_LEM.MAS

```
FILENAME=SAMP_LEM, SUFFIX=ADANAT  , $
  SEGMENT=SEG1, SEGTYPE=S0, $
$  GROUP=REC_BUFFER, USAGE=A49, ACTUAL=A28, $
    FIELDNAME=#REF_NAME, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_COUNT, USAGE=P4, ACTUAL=Z3, $
    FIELDNAME=#REF_FROM, USAGE=P16, ACTUAL=P8, $
    FIELDNAME=#REF_TO, USAGE=P16, ACTUAL=P8, $
    FIELDNAME=#REF_RC, USAGE=A1, ACTUAL=A1, $
  SEGMENT=SEG2, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_NAME, $
   GROUP=#REF_NAME, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_NAME_FIRST_2, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=#REF_NAME_LAST_6, USAGE=A6, ACTUAL=A6, $
  SEGMENT=SEG3, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_FROM, $
   GROUP=#REF_FROM, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#FCHAR, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG4, SEGTYPE=S0, PARENT=SEG3, OCCURS=8, POSITION=#FCHAR, $
    FIELDNAME=#FCHAR, USAGE=I9, ACTUAL=I1, $
  SEGMENT=SEG5, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_FROM, $
   GROUP=#REF_FROM, USAGE=A5, ACTUAL=A5, $
    FIELDNAME=#RESET_PARM, USAGE=A5, ACTUAL=A5, $
  SEGMENT=SEG6, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_FROM, $
   GROUP=#REF_FROM, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_FROM_A, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG7, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#TCHAR, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG8, SEGTYPE=S0, PARENT=SEG7, OCCURS=8, POSITION=#TCHAR, $
    FIELDNAME=#TCHAR, USAGE=I9, ACTUAL=I1, $
  SEGMENT=SEG9, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=#REF_TO_A, USAGE=A8, ACTUAL=A8, $
  SEGMENT=SEG10, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
   GROUP=#REF_TO, USAGE=A8, ACTUAL=A1, $
    FIELDNAME=#REF_TO_P1, USAGE=P2, ACTUAL=P1, $
```

```
SEGMENT=SEG11, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A8, ACTUAL=A2, $
  FIELDNAME=#REF_TO_P3, USAGE=P4, ACTUAL=P2, $
SEGMENT=SEG12, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A8, ACTUAL=A3, $
  FIELDNAME=#REF_TO_P5, USAGE=P6, ACTUAL=P3, $
SEGMENT=SEG13, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A8, ACTUAL=A4, $
  FIELDNAME=#REF_TO_P7, USAGE=P8, ACTUAL=P4, $
SEGMENT=SEG14, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A8, ACTUAL=A5, $
  FIELDNAME=#REF_TO_P9, USAGE=P10, ACTUAL=P5, $
SEGMENT=SEG15, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A8, ACTUAL=A6, $
  FIELDNAME=#REF_TO_P11, USAGE=P12, ACTUAL=P6, $
SEGMENT=SEG16, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A8, ACTUAL=A7, $
  FIELDNAME=#REF_TO_P13, USAGE=P14, ACTUAL=P7, $
SEGMENT=SEG17, SEGTYPE=U, PARENT=SEG1, OCCURS=1, POSITION=#REF_TO, $
 GROUP=#REF_TO, USAGE=A16, ACTUAL=A8, $
  FIELDNAME=#REF_TO_P15, USAGE=P16, ACTUAL=P8, $
```

Generated Access File SAMP_LEM.ACX

```
SEGNAME=SEG1, STPNAME=SAMPP001, OPTTRN=N, OPTPRM=C, OPTUPD=U,
  TRGDBID=3, TRGFILE=5, $
```

Reference: Master File Attributes

The following Master File attributes describe Adabas Stored Procedure data segments.

Attribute	Description
FILENAME	The Master File name. This name may or may not match the stored procedure name.
SUFFIX	Identifies the adapter, and is always ADANAT.
SEGMENT	The segments in the description that are created when the synonym is generated. The segment names follow a logical format to provide uniqueness within the file.
FIELDNAME	The field name from the data area.
GROUP	The fields from the data areas that were redefined or that were defined as arrays.

Attribute	Description
USAGE	The display format and length of the field. This attribute determines how the value is displayed in reports. Values are determined based on the format and length specified by the ACTUAL attribute.
ACTUAL	The format and length of the field as described in the data area.

Reference: Access File Attributes

Attribute	Description
SEGNAME	The name of the Master File segment that describes the stored procedure input parameters. If the stored procedure does not have input parameters, the synonym generation process creates a dummy segment.
STPNAME	The name of the stored procedure. You specify the value when you create the stored procedure synonym.
TRGDBID	The database ID of the trigger. You specify the value when you create the stored procedure synonym.
TRGFILE	The name of the trigger file. You specify the value when you create the stored procedure synonym.
OPTTRN	The transaction option. You specify the value when you create the stored procedure synonym.
OPTPRM	The parameter option. You specify the value when you create the stored procedure synonym.
OPTUPD	The update option. This is based on values that you specify when you create the stored procedure synonym.

Invoking an Adabas Stored Procedure

To invoke an Adabas stored procedure, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, or an SQL SELECT statement. (DML is Information Builders internal retrieval language.) For information about the Data Manipulation Language, see the *Stored Procedure Reference* manual.

Syntax: How to Invoke an Adabas Stored Procedure Using TABLE

The syntax for invoking an Adabas stored procedure using a TABLE command is

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the Adabas stored procedure you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

IF/WHERE

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke an Adabas Stored Procedure Using SELECT

The syntax for invoking an Adabas stored procedure using a SELECT statement is

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the Adabas stored procedure you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

WHERE

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke an Adabas Stored Procedure Using EX

```
EX [app_name_space/]synonym  1=parm1_val,..., N=parmN_val

EX [app_name_space/]synonym parm1_name=parm1_val,... ,
                            parmN_name=parmN_val

EX [app_name_space/]synonym [, parm1_val [,...[, parmN_val]]]
```

where:

app_name_space

Is the apps directory name under which the synonyms are stored. This value is optional.

synonym

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

parm_name

Is the name of the parameter taken from the input metadata description.

1=parm1_val

N=parmN_val

parm1_name

parm1_val

parmN_val

Are the parameter values that match the input metadata description.

Note:

- Consecutive commas denote missing parameters in the positional form of the request.
- Values containing special characters (<equal sign> <space> <comma>) must be encapsulated in double or single quotation marks.

Example: Invoking the SAMP_LEM Adabas Stored Procedure

The following TABLE command invokes the SAMP_LEM Adabas stored procedure, passing three parameter values to it.

```
TABLE FILE SAMP_LEM
   PRINT #REF_RC SEG4.#FCHAR
   IF #REF_NAME EQ '12345678'
   IF #REF_COUNT EQ 123
   IF #RESET_PARM EQ 'RESET'
END
```

The following SELECT command invokes the SAMP_LEM Adabas stored procedure:

```
SQL
SELECT #REF_RC SEG4.#FCHAR FROM SAMP_LEM
   WHERE #REF_COUNT = 123
END
```

The following EX commands invoke the SAMP_LEM Adabas stored procedure:

```
EX SAMP_LEM 225,2004
EX SAMP_LEM AM=225,YEAR=2004
EX SAMP_LEM 1=225,2=2004
```

Chapter 5

Using the Adapter for Alchemy Sentiment Analysis

This section describes how to configure the Alchemy Sentiment Analysis Adapter.

In this chapter:

Alchemy Sentiment Analysis Adapter Overview

The Alchemy Sentiment Analysis Adapter is used to score structured and unstructured textual content by identifying positive, neutral, and negative sentiment found within emails, documents, and database records. Textual data from a data source is passed to the adapter in one of three ways:

- Joining the column containing the textual data from the data source to the column within the Alchemy Sentiment Analysis Adapter used to define the textual data to be scored.
- A report which uses Cluster Join metadata. The Cluster Join metadata already contains the join from the column containing the textual data from the data source to the column within the Alchemy Sentiment Analysis Adapter used to define the textual data to be scored.
- Using a WHERE/IF condition to pass textual data directly to the column within the Alchemy Sentiment Analysis Adapter used to define the textual data to be scored.

The score returned from the Alchemy Sentiment Analysis Adapter ranges from -1 to 1.

- A score of -1 identifies the sentiment of the textual data that was passed to the adapter as extremely negative.
- A score of 0 identifies the sentiment of the textual data that was passed to the adapter as neutral.
- A score of 1 identifies the sentiment of the textual data that was passed to the adapter as tremendously positive.

Configuring the Alchemy Sentiment Analysis Adapter

The Alchemy Sentiment Analysis Adapter is part of the Social Media group of adapters within the WebFOCUS Reporting Server.

Procedure: How to Configure the Alchemy Sentiment Analysis Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

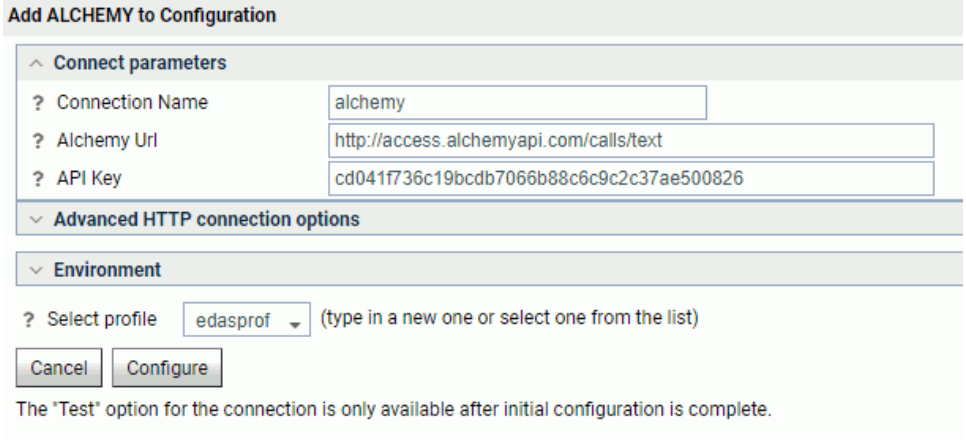
   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click *ALCHEMY* and select *Configure*.

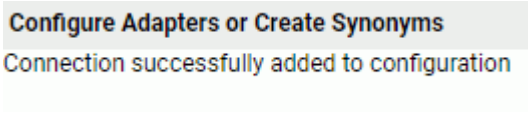
   The Add ALCHEMY to Configuration pane opens, as shown in the following image.



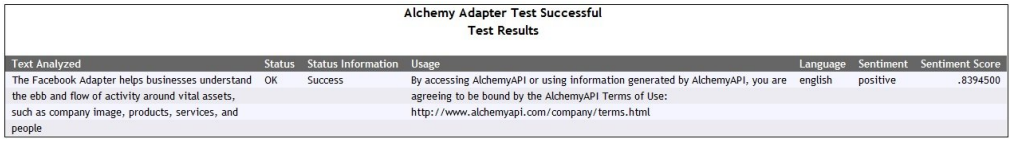
5. Enter the value for API Key supplied by Alchemy.

6. Click *Configure*.

The Configure Adapters or Create Synonyms pane opens, as shown in the following image.



7. Click *Test* to ensure that the Alchemy Sentiment Analysis Adapter is configured properly.



Alchemy Adapter Test Successful
Test Results

Text Analyzed	Status	Status Information	Usage	Language	Sentiment	Sentiment Score
The Facebook Adapter helps businesses understand the ebb and flow of activity around vital assets, such as company image, products, services, and people	OK	Success	By accessing AlchemyAPI or using information generated by AlchemyAPI, you are agreeing to be bound by the AlchemyAPI Terms of Use: http://www.alchemyapi.com/company/terms.html	english	positive	.8394500

Reference: Connection Attributes for Alchemy Sentiment Analysis

The following list describes the connection attributes for the Alchemy Sentiment Analysis Adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Alchemy URL

The URL that is used to connect to the Alchemy Sentiment Analysis service. The default value is:

`http://access.alchemyapi.com/calls/text`

API KEY

The API Key that is supplied by Alchemy to allow authorization to the Alchemy Sentiment Analysis scorer.

PROXY Server IP Address

IP address of the proxy server. For example:

`170.115.249.42`

PROXY Port

Port number on which the proxy server listens. The default port number is 80.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the connection attributes. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), choose New Profile from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

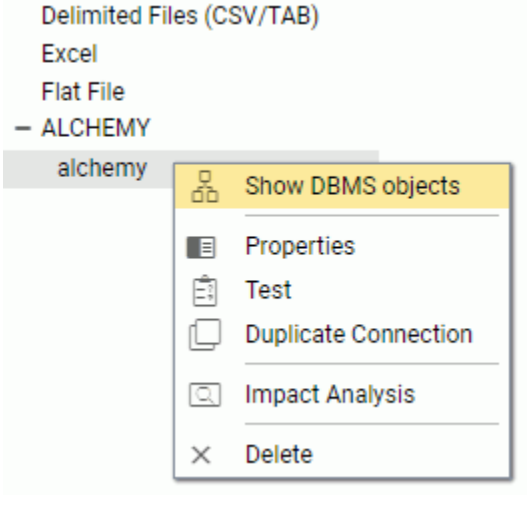
Store the connection attributes in the server profile (edasprof).

Creating Metadata and Sample Reports for the Alchemy Adapter

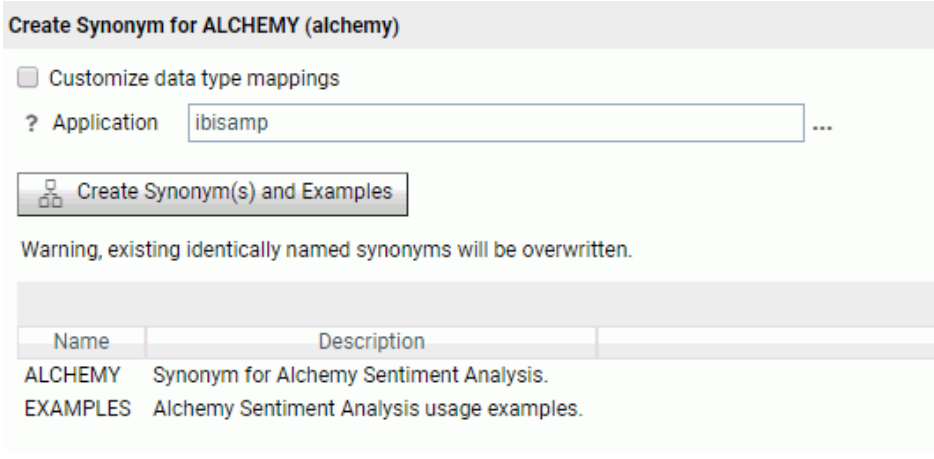
Create Synonym for the Alchemy Sentiment Analysis Adapter creates the metadata used by reports to perform Sentiment Analysis scoring as well as sample reports which utilize the metadata.

Procedure: How to Create Metadata and Sample Reports

1. From the WebFOCUS Reporting Server Web Console or the Data Management Console, expand the *Adapters* folder, *Configured* folder, and then the *ALCHEMY* folder.
2. Right-click the configured connection for the Alchemy Sentiment Analysis Adapter (for example, alchemy) and select *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for ALCHEMY pane opens, as shown in the following image.

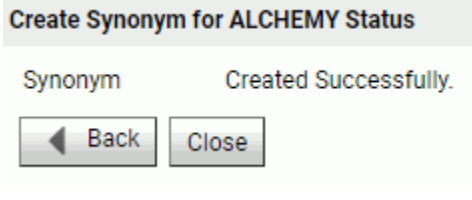


3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata and sample reports are to be stored.

   The sample reports are stored within the *alchsampl* subdirectory of the application.

4. Click *Create Synonym(s) and Examples*.

   The Create Synonym for ALCHEMY Status pane opens and indicates that the synonym was created successfully, as shown in the following image.



Alchemy Sentiment Analysis Adapter Examples

This section describes the metadata and sample reports for the Alchemy Sentiment Analysis Adapter.

Reference: Alchemy Sentiment Analysis Adapter Metadata

The following table lists and describes the available metadata for the Alchemy Sentiment Analysis Adapter.

Metadata	Description

alchemy	Metadata is used for interacting with the Alchemy web service for sentiment analysis scoring.
	
	The DOC field would contain the textual data that is to be analyzed and scored by the Alchemy Sentiment scorer. Textual data can be joined from a column within a table to the DOC field or set within a WHERE/IF condition.
	
	The following example uses a JOIN statement:
	
	`JOIN DOCLINE IN alchemy/alchsampl/ alchemy_sample_fix TO DOC IN alchemy/ alchemy END`
	
	The following example uses a WHERE/IF condition:
	
	`WHERE (DOC CONTAINS 'The Facebook Adapter helps businesses')`
	
	A sentiment score between -1 and 1 is returned within the SCORE field.

alchsampl/alchemy_sample_fix	Metadata that defines the sample text file (alchsampl/alchemy_sample.txt) used for the *alchemy_sample_join* and *alchemy_sample_cluster* sample reports.
alchsampl/alchemy_sample_cluster	Cluster join between *alchsampl/alchemy_sample_fix* and *alchemy*.

Reference: **Alchemy Sentiment Analysis Adapter Sample Reports**

The following table lists and describes the sample reports for the Alchemy Sentiment Analysis Adapter.

Sample Report	Description
alchsampl/alchemy_sample_join	Scores the text passed from the *alchsampl/alchemy_sample.txt* file performed through a JOIN statement.
alchsampl/alchemy_sample_cluster	Scores the text passed from the *alchsampl/alchemy_sample.txt* file using the Cluster Join master.
alchsampl/alchemy_sample_where	Scores the text passed within a WHERE statement.

Chapter 6

Using the Adapter for Amazon Athena

The adapter for Amazon Athena allows applications to access and analyze data stored in the Amazon Simply Storage Server (Amazon S3) using standard SQL.

In this chapter:

Introducing the Adapter for Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. You can point Athena at your data stored in Amazon S3, run a SQL or ad-hoc query and get results fast.

The Adapter for Amazon Athena is available in two types: ODBC and JDBC. Only one type per server instance can be configured. Amazon provides ODBC (on Windows and Linux) and JDBC drivers and adapter of corresponding type using ODBC or JDBC Athena API to connect and report.

Preparing the Amazon Athena Environment

The following components are needed to use the adapter for Amazon Athena:

ODBC Driver:

- Download and install the Amazon Athena ODBC driver.
- Perform the configuration steps according to the Amazon/Simba instructions for your operating system.
- Obtain the AWS Region, S3 Output Location, Encryption Option, AWS access key, and AWS secret key from your Amazon Athena Administrator.

JDBC Driver:

- Download and install the Amazon Athena JDBC driver.
- Obtain the path to the JDBC driver jar files that you will use. For example, using JDBC 4.1 that is Java 8 compatible on Linux requires:

  ```
  /qas/authena/AthenaJDBC41-1.0.1.jar
  ```

  This full path to the Athena driver .jar files value will be added to IBI_CLASSPATH during configuration.
- Obtain the connection URL, S3 staging directory, AWS access key and AWS secret key from your Amazon Athena Administrator.

Configuring the Adapter for Amazon Athena

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Amazon Athena ODBC Adapter Configuration Settings

The Adapter for Amazon Athena ODBC is under the SQL group folder. The configuration settings for the adapter are:

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

DSN-less

Select this option to provide AWS Region and S3 Output Location in lieu of a DSN.

DSN

The DSN that was configured for your Athena Data Source.

Security

Select *Explicit* as the authentication method to use when connecting to the database server.

User

Enter the AWS access key as the primary authorization ID.

Password

Enter the AWS secret key as the password associated with the primary authorization ID.

Reference: Amazon Athena JDBC Adapter Configuration Settings

The Adapter for Amazon Athena is under the SQL group folder. The configuration settings for the adapter are:

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the location of the Amazon Athena JDBC data source.

The following is an example of the URL that is used to connect to your Amazon Athena server:

```
jdbc:awsathena://connection_URL:?s3_staging_directory
```

where:

connection_URL

Is the connection URL obtained by your Amazon Athena administrator.

s3_staging_directory

Is the staging directory obtained by your Amazon Athena administrator.

Security

Select *Explicit* as the authentication method to use when connecting to the database server.

User

Enter the AWS access key as the primary authorization ID.

Password

Enter the AWS secret key as the password associated with the primary authorization ID.

Driver Name

Enter *com.amazonaws.athena.jdbc.AthenaDriver* as the name of the JDBC driver.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Enter the full path to the Athena driver jars, as determined in *Preparing the Amazon Athena Environment* on page 265.

Creating Synonyms With Amazon Athena

Synonyms define unique names (or aliases) for each Amazon Athena external table (External Table is the only object type that is available in Athena) that is accessible using a connection. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Athena external table.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.
5. Optionally, expand the *Customize data type mappings* section and enter a value for Longchar Length or numeric precision.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Limitations

Athena uses an approach known as schema-on-read, which means a schema is projected on to your data at the time that you execute a query. This eliminates the need for data loading or transformation. Athena does not modify your data in Amazon S3, but instead uses Apache Hive to define tables and create databases, which are essentially a logical namespace of tables.

When you drop a table in Athena, only the table metadata is removed; the data remains in Amazon S3.

Chapter 7

Using the Adapter for Amazon Redshift

The Adapter for Amazon Redshift allows applications to access Amazon Redshift data sources. The adapter converts data or application requests into native Amazon Redshift SQL statements and returns optimized answer sets to the requesting program.

In this chapter:

Introducing the Adapter for Amazon Redshift

The Adapter for Amazon Redshift is ODBC based. It utilizes the Amazon Redshift ODBC driver, which can be installed on Windows and Linux.

Preparing the Amazon Redshift ODBC Environment

On Windows, the Amazon Redshift ODBC environment is set up during the installation of the Redshift ODBC driver. No additional setup steps are required.

On Linux, after the Amazon Redshift ODBC driver is installed, make sure that path to the directory where the Redshift driver is installed is included in LD_LIBRARY_PATH (or in IBI_LIBPATH if the server is running with security on).

For example:

```
export IBI_LIBPATH=/opt/amazon/redshiftodbc/lib/64
```

Also, set variable AMAZONREDSHIFTODBC that points to the Linux driver manager initialization file with the setting DriverManagerEncoding=UTF-16 in it.

For example:

```
export AMAZONREDSHIFTODBC=/usr/odbc/ibiodbc/amazon.redshiftodbc.ini
```

Configuring the Adapter for Amazon Redshift

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Amazon Redshift Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Amazon Redshift Adapter Configuration Settings

The Adapter for Amazon Redshift is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

System

Is the name of the machine where the Amazon Redshift database is running. The basic syntax is *host*. Note that the port can be specified in the Additional connection string keywords text box.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Additional connection string keywords (optional)

Is used to add options to the connection string. For example, 'PORT=1563'.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Creating Synonyms With Amazon Redshift

Synonyms define unique names, or aliases, for each Amazon Redshift table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Amazon Redshift table.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Using Direct Pass-through With Amazon Redshift

Direct Pass-through commands, such as SHOW TABLES and DESCRIBE tablename, do not display any results.

This can be avoided by adding a comment:

```
SQL SQLRDS /*QUERY*/ show all;
END
```

Chapter 8

Using the Adapter for Apache Drill

Apache Drill is a query engine for Hadoop and a variety of NoSQL databases and file systems including HDFS and Mapr-FS, HBase and MapR-DB, and MongoDB. Drill can use Hive metadata or derive metadata from self-describing data files.

In this chapter:

Preparing the Apache Drill Environment

The following components are needed to use the adapter for Apache Drill:

- **Java.** To use this Java-based adapter, you must have Java installed. Drill requires Version 1.7 or later. Java can be downloaded from *http://www.java.com*.

  The location of the Java must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/lib/jvm/jre-1.7.0
  ```

  If you have JDK installed:

  ```
  export JAVA_HOME=/usr/lib/jvm/jdk-1.7.0
  ```

  If you are using Windows, right-click *Computer* and select *Properties*. Then select *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

  ```
  C:\Program Files\Java\jdk7\bin\server;C:\Program Files\Java\jdk7\bin;
  ```

- **JDBC Drivers**

  There are two JDBC drivers available for Apache Drill.

  The Open Source JDBC Driver is included with Drill. It consists of a single file that can be found after installation at the following location:

```
$DRILL_HOME/jars/jdbc-driver/drill-jdbc-all-1.12.0.jar
```

It can also be downloaded from *https://drill.apache.org/docs/using-the-jdbc-driver*.

If you are using the Open Source JDBC Driver and are installing the server on the same system where Drill is installed, you can point to the jar files as described in the next section. If you are installing the server on some other system, copy those files to a location of your choice.

The MapR JDBC driver is available from MapR, as explained in step 3 of the following procedure. This driver also supports MapR SASL authentication.

Procedure: How to Configure the Java CLASSPATH

The location of the Drill jar files must be specified to the server. If you are running the server on the same system as a drillbit, you can specify its location. If you are running the server on another system, copy the files listed below to some location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console menu bar, select *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.

2. Expand the *Java Services* folder, then right-click *DEFAULT*, and select *Properties*.

   The Java Services Configuration page opens.

3. Click on the chevrons to expand Class Path.

   For the Open Source JDBC Driver

   In the IBI_CLASSPATH box, enter the full location of the jar file. For example:

   ```
   /opt/drill/apache-drill-1.12.0/jars/jdbc-driver/drill-jdbc-
   all-1.12.0.jar
   ```

   If you want to access data stored in HBase or MapR-DB, add an additional jar file. For example:

   ```
   /opt/drill/apache-drill-1.12.0/jars/3rdparty/hadoop-common-2.7.0.jar
   ```

   Note: The file names must be entered one per line.

If you are installing the adapter on a different system than where you have Drill installed, copy the jar files to a location on that system.

Note: For a server running on Windows, use Windows syntax for directory names. For example:

```
C:\Drill\jars\jdbc-driver\drill-jdbc-all-1.12.0.jar
```

For the MapR (Simba) JDBC Driver

Download the The MapR JDBC Driver from *https://maprdocs.mapr.com/home/Drill/drill_jdbc_connector.html* from the link for the Drill JDBC Driver provided by MapR. This will download a file named:

```
MapRDrillJDBC41-1.5.8.1017.zip
```

Note: The numbers will change with new releases.

Unzip the file to a location on your server. The 46 jar files constitute the MapR Drill JDBC Driver.

Add the names of all the jar files to your CLASSPATH or IBI_CLASSPATH, as described above.

4. Scroll down and click the *Save and Restart Java Services* button.

Configuring the Adapter for Apache Drill

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Apache Drill Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

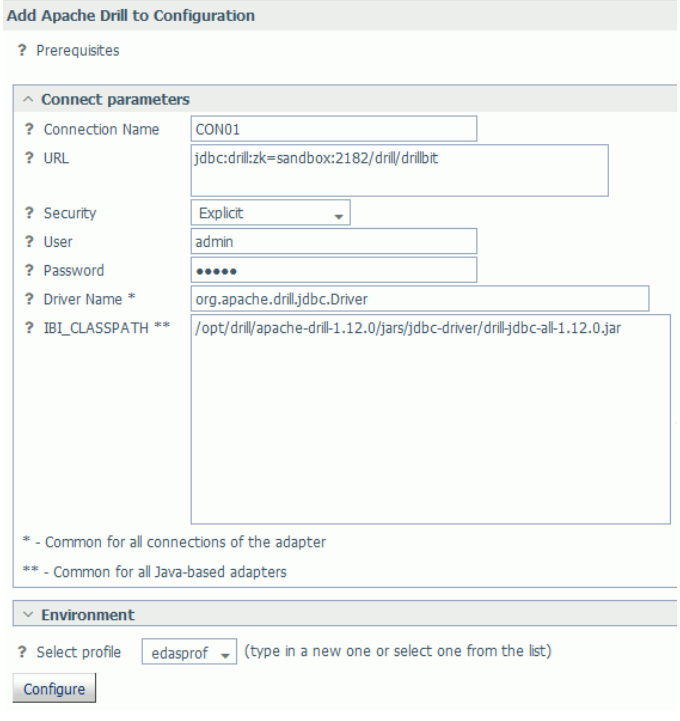
   The Add Adapter to Configuration pane opens.
5. In the URL box, enter the URL used to connect to your Drillbit. For more information, see *Apache Drill URL Configuration Settings*.
6. In the Driver Name box, type the JDBC driver that you are using from the following table:

   `org.apache.drill.jdbc.Driver`

Adapter	JDBC Driver Name
Apache Hive	`org.apache.drill.jdbc.Driver`
MapR Drill	`com.mapr.drill.jdbc41.Driver`

7. Enter your User and Password.

   The following image shows an example of the configuration settings used:



8. Click *Test*. You should see a list of data sources on your server.
9. Click *Configure*.

Reference: Connection Attributes for Apache Drill

The Adapter for Apache Drill is under the SQL group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Apache Drill data source.

Driver Name

Name for the Apache Drill driver.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Reference: Apache Drill URL Configuration Settings Using the Open Source rill JDBC Driver

The Adapter for Apache Drill is under the SQL group folder.

The URL configuration for the Apache Drill adapter is:

`jdbc:drill:zk=`*`quorum`*`:`*`port`*`/`*`directory`*`/`*`cluster_id`*`[;schema=`*`schema`*`]`

where:

`quorom`

Is the name or IP address where the Drillbit server is running.

`port`

Is the port number for zookeeper. The default value is 2181. On a MapR cluster, the default value is 5181.

Multiple quorum:port values can be comma separated.

`directory`

Is the name of the drill directory within zookeeper. The default value is drill.

`cluster_id`

Is the name of the drill cluster. The default value is drillbit1.

`schema`

Is the name of the default schema to use, for example, hive.

Reference: Apache Drill URL Configuration Settings Using the MapR JDBC Driver

The Adapter for Apache Drill is under the SQL group folder.

The URL configuration string for a client to Drillbit connection is:

```
jdbc:drill:drillbit=host[:port][:auth=method[:delegationuid=id]
```

where:

host

Is the name or IP address where the drillbit is running.

port

Is the drillbit port. The default value is 31010.

The URL configuration string for a client to Zookeeper connection is:

```
jdbc:drill:zk=quorom[:port]/drill/[cluster]
[:auth=method[:delegationuid=id] ]
```

where:

quorom

Is the name or IP address where Zookeeper is running.

port

Is the Zookeeper port. The default value is 5181 for a MapR cluster.

cluster

Is the name of the drillbit cluster.

For either connection:

method

Is the authentication method. Values are plain, kerberos, or maprsasl.

id

Specifies the user ID it is connected to, when the authentication method is maprsasl.

Troubleshooting

If the server is unable to configure the connection, a dialog box opens with an error message. The message typically begins with the following:

```
(FOC1400) SQLCODE IS -1 (HEX: FFFFFFFF) XOPEN: nnnn
```

where:

nnnn

Is the message number returned.

Some of the more common errors are:

```
(FOC1500) : (-1) [00000] JDBFOC>> connectx():
(FOC1500) : java.lang.UnsupportedClassVersionError:
org/apache/drill/jdbc/Driver :
(FOC1500) : Unsupported major.minor version 51.0
(FOC1479) ERROR CONNECTING TO SQL DATABASE
```

This indicates that your Java version is less than 1.7 which is required for Drill.

Creating Synonyms With Apache Drill

Synonyms define unique names, or aliases, for each Hive table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Hive table.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

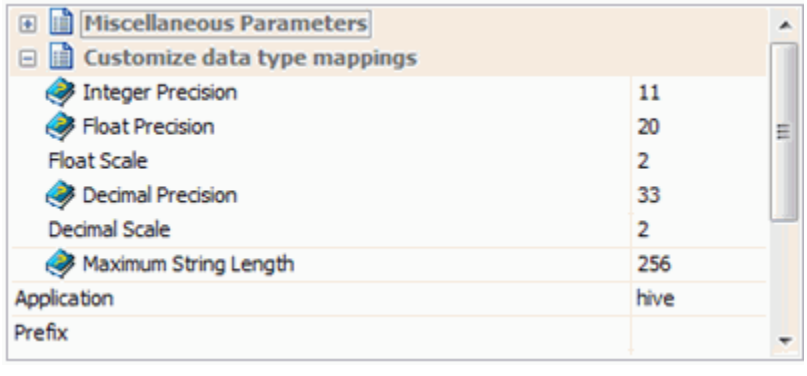
   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.
5. Optionally, expand the *Customize data type mappings* section and enter a value for Maximum String Length or numeric precision, as shown in the following image.



6. Select the check box for the objects that you want to create synonyms for. If you want to change the name of the synonym from the default name, click the name and edit it as needed.
7. Click *Create Synonym*.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Reference: Synonym Creation Parameters for Apache Drill

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Chapter 9

Using the Adapter for Apache Hive to Access Data Managed by Hadoop

The adapter for Hadoop provides for analysis of both structured and complex data. Hive provides a JDBC driver and the Hive Query Language, a SQL-like interface for generating MapReduce programs.

In this chapter:

Introducing the Adapter for Apache Hive

Hadoop is a collection of open source products and technologies administered by the Apache Software Foundation. It provides for analysis of both structured and complex data. The core components of Hadoop are Hadoop DFS (Distributed File System) and the MapReduce programming framework. Hadoop is available from Apache, as well as many commercial and community distributions. The leading vendors are Cloudera, Hortonworks, and MapR.

Hadoop is designed to run on Linux and is available from many distributors. There is a Windows version available from Microsoft and Hortonworks that runs on Windows Server 2012 R2.

Hive provides a JDBC driver and the Hive Query Language, a SQL-like interface for generating MapReduce programs. While Hive was originally designed for batch processes, it can now be used for reporting and Business Intelligence. For use with WebFOCUS, Hive 1.0 or later is recommended.

Some database vendors provide their own Hadoop distributions with their proprietary SQL engines. This includes IBM Big SQL, MS SQL Server PolyBase, and others. If you are using one of these vendor products in addition to using the Apache Hive adapter, you can also use the Information Builders data adapter for the appropriate product.

The Hive adapter is JDBC based. The DataMigrator or WebFOCUS server can be run on any platform (including Windows) that connects to the server where Hive is running.

Preparing the Apache Hive Environment

The following components are needed to use the adapter for Hadoop/Hive:

- **Java.** To use this JDBC-based adapter, you must have Java installed. Version 1.8 or later is recommended. Java can be downloaded from *http://www.java.com*.

  The location of Java must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/lib/jvm/jre-1.8.0
  ```

  If you have JDK installed:

  ```
  export JAVA_HOME=/usr/lib/jvm/jdk-1.8.0
  ```

  If you are using Windows, right-click *Computer* and select *Properties*. If using Windows 10, click *Settings* in the Start menu and search for *Environment*. Click *Edit the system environment variables* to open the System Properties dialog box. Click the *Advanced* tab and click *Environment Variables*. Add the locations to your PATH variable. For example:

  ```
  C:\Program Files\Java\jdk8\bin\server;C:\Program Files\Java\jdk8\bin;
  ```

- **JDBC Drivers.** The Apache JDBC driver is distributed as a jar file, which is included in a Hadoop distribution. If you are installing the server on a node or an edge node of your Hadoop cluster, you can point to the hive-jdbc-<*version*>-standalone.jar file as described in the next section. If you are installing the server on some other system, copy the file to a location of your choice.

  If you are using the Cloudera/Simba JDBC driver, download the driver as described in step 3 below.

Procedure: How to Configure the Java CLASSPATH

The location of the Hadoop and Hive jar files must be specified to the server. If you are running the server on a node or an edge node of your Hadoop cluster, you can specify their location. If you are running the server on another system, copy the files listed below to some location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console menu bar, select *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.

2. Expand the *Java Services* folder. Right-click *DEFAULT* and click *Properties*.

   The Java Services Configuration page opens.

3. Expand *Class path*.

   In the IBI_CLASSPATH box, enter the full location of the Hive and Hadoop files shown below, where hive_home is where Hive is installed and hadoop_home is where Hadoop is installed. You must type them explicitly and cannot use $HIVE_HOME. The file names must be entered one per line.

   If you are installing the adapter on a different system than where Hadoop and Hive are installed, copy the jar files to a location on that system.

   Note: For a server running on Windows, use Windows syntax for directory names. For example:

   ```
   C:\jdbc\hive-jdbc-<version>-standalone.jar
   ```

   For Apache Hive:

   Enter the full path to the location of the JDBC client jar on your system:

   ```
   hive-jdbc-<version>-standalone.jar
   ```

   The following jar files may also be required when using the HTPP transport, connecting to a Kerberos enabled cluster, or using HBase as a source.

   ```
   hadoop-common.jar
   hadoop-auth.jar
   ```

   For Cloudera/Simba:

   1. Download the JDBC driver from Cloudera from the Downloads page at *http://www.cloudera.com/downloads.html*.
   2. From the Database Drivers section of the page, click the *Hive JDBC Driver Downloads* link.
   3. Select your operating system and version from the drop-down menus, and click *Get It Now*.

This downloads a file called hive_jdbc_2.5.20.zip, which contains two .zip files. (Note that these numbers will change with each new release.) Unzip the Cloudera_HiveJDBC41_2.5.20.*nnnn*.zip to a location on your system.

Note: The 41 in the file name indicates JDBC 4.1, which supports Java 1.7 or later.

These 15 jar files comprise the Cloudera/Simba JDBC driver. Add the names of all of them to your CLASSPATH or IBI_CLASSPATH, as described above in this step.

4. Scroll down and click the *Save and Restart Java Services* button.

Configuring the Adapter for Hadoop/Hive

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The adapter supports Hiveserver2, which is available in Hive 0.11.0 and later. It is strongly recommended for improved throughput, functionality, and security.

Procedure: How to Configure the Hadoop/Hive Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

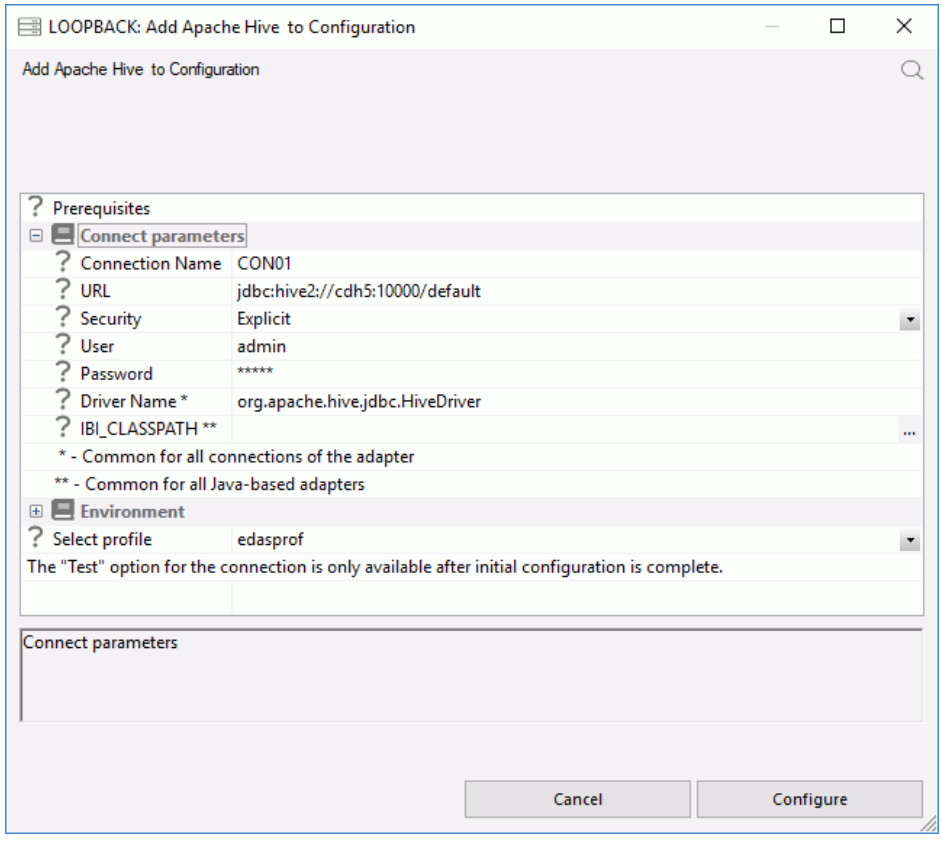
   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click *Apache Hive* and click *Configure*.
5. In the URL box, type the URL used to connect to your Hive server. For more information, see *Hive/Impala Adapter Configuration Settings*.
6. In the Driver Name box, type the driver name from the following table:

Driver	JDBC Driver Name
Apache Hive	`org.apache.hive.jdbc.HiveDriver`
Cloudera/Simba	`com.cloudera.hive.jdbc41.HS2Driver`

7. Select the security type. If you are using Explicit, type your user ID and password.

   The following image shows an example of the configuration settings used:



8. Select *edasprof* from the Select profile drop-down menu to add this connection for all users, or select a user profile.
9. Click *Test*. You should see a list of data sources on your server.
10. Click *Configure*.

Reference: Hive Adapter Configuration Settings

The Adapter for Hadoop/Hive is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the location of the data source.

The URL used depends on what type of server you are connecting to. See the table below for examples. For more details, see the JDBC section in the online documentation for Apache Hive, or the *Cloudera JDBC Driver for Apache Hive* documentation, which is included with the JDBC driver download.

Server	URL
Hiveserver2	`jdbc:hive2://server:10000/default`
Kerberos Hiveserver2 (static)	`jdbc:hive2://server:10000/default; principal=hive/server@REALM.COM`
Kerberos Hiveserver2	`jdbc:hive2://server:10000/default;principal=hive/server@REALM.COM; auth=kerberos;kerberosAuthType=fromSubject`
Cloudera/Simba (no security)	`jdbc:hive2://server:10000;AuthMech=3;transportMode=binary`
Kerberos Cloudera/Simba	`jdbc:hive2://server:10000;AuthMech=1;KrbRealm=REALM.COM; KrbHostFQDN=server.example.com;KrbServiceName=hive`

where:

server

Is the DNS name or IP address of the system where the Hive server is running. If it is on the same system, localhost can be used.

default

Is the name of the default database to connect to.

`10000`

Is the default port number HiveServer2 is listening on if not specified when the Hive server is started.

`REALM.COM`

For a Kerberos enabled Hive server, this Is the name of your realm.

Driver Name

Is the name of the JDBC driver, for example, org.apache.hive.jdbc.HiveDriver or com.cloudera.hive.jdbc41.HS2Driver.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The user ID is passed from a trusted source.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention). Choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Kerberos

Connections to a Hive server with Kerberos enabled can be run in several ways:

- **Static.** The same Kerberos credential is used for all connections. You must obtain a Kerberos ticket before starting the server. On expiration of the Kerberos ticket, a new one must be obtained and the server must be restarted. Use *Trusted* as the security type.

  This mode is useful for testing, but is not recommended for production deployment.

- **User.** Each user connecting to the server connects to Hive using their own credentials from the Hive Adapter connection in the user profile. The server obtains a Kerberos ticket for the user. Use *Explicit* as the security type.

- **Database.** Users connect to the server using the Kerberos credentials, which are authenticated against a Hive server with Kerberos enabled. The server obtains a Kerberos ticket for the user. Use *Password Passthru* as the security type.

- **Single Sign-on.** Users log into Windows with their Kerberos or Active Directory credentials and Windows obtains a Kerberos ticket. That ticket is used when the user connects to the server and is passed to Hive. Use *Trusted* as the security type.

To setup connections to a Kerberos enabled Hive instance:

- The Reporting Server has to be secured. The server can be configured with security providers PTH, LDAP, DBMS, OPSYS, or Custom, as well as multiple security providers environment,

Kerberos Static Ticket Requirements

In this configuration, all connections to Hive instance will be done with the same Kerberos user ID derived from the Kerberos ticket that is created before the server starts.

1. Create Kerberos ticket using:

   ```
   kinit kerbid01
   ```

where:

kerbid01

Is a Kerberos ID.

2. Verify Kerberos ticket using *klist*. The following message should be returned:

```
Ticket cache: FILE:/tmp/krb5cc_532
Default principal: kerbid01@REALM.COM

Valid starting     Expires            Service principal
04/29/16 16:26:50  04/30/16 02:26:53  krbtgt/REALM.COM@REALM.COM
    renew until 05/06/16 16:26:50
```

3. Before configuring the Hive Adapter connection to a Kerberos-enabled instance, the connection should be tested. Log in to the system running Hive and use Beeline, the native tool, to test it.
4. Start the server in the same Linux session where the Kerberos ticket was created. Log in to the Web Console and click the *Adapters* tab.
5. Right-click *Apache Hive*. Use the following parameters to configure the adapter:

URL

Enter the same URL that you use to connect to the Hive Instance using Beeline. For example:

```
jdbc:hive2://server:10000/default;principal=hive/server@REALM.COM
```

Security

Set to Trusted.

6. In the Select profile drop-down menu, select the *edasprof* server profile.
7. Click *Configure*.
8. Next, configure Java services. Click the *Workspace* tab and expand the *Java Services* folder.
9. Right-click *DEFAULT* and select *Properties*.
10. Expand the *JVM Settings* section. In the JVM options box, add the following:

```
-Djavax.security.auth.useSubjectCredsOnly=false
```

11. Restart Java services.

Once these steps are completed, the adapter can be used to access a Kerberos-enabled Hive instance.

Kerberos User Ticket Requirements

In this configuration, each connected user has a Hive Adapter connection with Kerberos credentials in the user profile.

1. Enable multi-user connection processing for Kerberos by adding the following line to your profile (edasprof.prf):

   ```
   ENGINE SQLHIV SET ENABLE_KERBEROS ON
   ```

2. Kerberos looks for the default realm in /etc/krb5.conf, or on Windows krb5.ini. Make sure that this file contains the information for your Kerberos server, and not the sample file.

   If you are running the server on a Windows system, under the Local System account, specify your realm name and Kerberos Key Distribution Center. To do so:

   a. Expand the *Workspace* folder.

   b. Expand the *Java Services* folder, right-click *DEFAULT*, and click *Properties*.

   c. Expand the JVM settings section and add the following in the JVM Options box:

      ```
      -Djava.security.krb5.realm=REALM.COM
      -Djava.security.krb5.kdc=kdc.realm.com
      ```

      where:

      REALM.COM
      Is the realm name of your organization.

      kdc.realm.com
      Is the Key Distribution Center of your organization.

3. Configure the Hive Adapter Connection in the user profile using the following values:

   URL
   Is the URL to the location of the data source.

Server	URL
Kerberos Hiveserver2 (User)	`jdbc:hive2://server:10000/default;principal=hive/server@REALM.COM; auth=kerberos;kerberosAuthType=fromSubject`

Server	URL
Kerberos Cloudera/Simba	`jdbc:hive2://server:10000;AuthMech=1;KrbRealm=REALM.COM;KrbHostFQDN=server.example.com;KrbServiceName=hive`

Security

Set to Explicit.

User and Password

Enter your Kerberos user ID and password. The server will use those credentials to create a Kerberos ticket and connect to a Kerberos-enabled Hive instance.

Note: The user ID that you use to connect to the server does not have to be the same as the Kerberos ID you use to connect to a Kerberos-enabled Hive instance.

Select Profile

Select your profile or enter a new profile name consisting of the security provider, an underscore and the user ID. For example, *ldap01_pgmxxx*.

4. Click *Configure*.

Kerberos Password Passthru Requirements

This procedure will show you how to create a connection to a Kerberos enabled Hadoop Cluster using the Adapter for Apache Hive with security Password Passthru.

1. Configure the connection to Hive using the following values:

   URL

   Is the URL to the location of the data source.

   ```
   jdbc:hive2://server1:10000/default;principal=hive/server1@REALM.COM;
   auth=kerberos;kerberosAuthType=fromSubject
   ```

   where:

   server1

   Is the name of a node on your Hadoop cluster where a Hive2 server is running.

   Security

   Set to Password Passthru.

   Select Profile

   Select the server profile, *edasprof*.

   Driver Name

   Is the name of the JDBC driver, for example, org.apache.hive.jdbc.HiveDriver.

   IBI_CLASSPATH

   Defines the additional Java Class directories or full-path jar names which will be available for Java Services (if not included in your system CLASSPATH).

   ```
   /path/to/hive-jdbc-<version>standalone.jar
   ```

2. Click *Configure*.

   You also need to configure Hive as a DBMS Security Provider.

3. From the Web Console, click *Access Control*.
4. Under Security Providers, right-click *DBMS* and click *New*.

   The DBMS Security Provider Configuration page opens.

5. Enter the following values:

   DBMS Provider

   Enter the provider, as desired. For example, hive.

   Security DBMS

   Select *SQLHIV - Apache Hive*.

   security_connection

   The connection name you specified, for example, *server*.

6. Click *Save*.

   The DBMS Security Provider Configuration page opens.

7. Select the line *DBMS - hive* , where *hive* is the name you used. .
8. From the Status drop-down menu, select *Primary* or *Secondary*, as desired.

Kerberos Single Sign-On Requirements

The following sections will show you how to configure a DataMigrator or WebFOCUS Reporting Server for access to Hadoop through Hive with Kerberos enabled Hadoop Cluster using the Adapter for Apache Hive with single sign-on from the desktop to the browser using Hive.

1. Configure the connection to Hive using the following values:

   URL

   Is the URL to the location of the data source.

   ```
   jdbc:hive2://server1:10000/default;principal=hive/server1@IBI.COM;
   auth=kerberos;kerberosAuthType=fromSubject
   ```

   where:

   server1

   Is the name of a node on your Hadoop cluster where a Hive2 server is running.

   Security

   Set to Trusted.

   Select Profile

   Select the server profile, *edasprof*.

   Driver Name

   Is the name of the JDBC driver, for example, org.apache.hive.jdbc.HiveDriver.

   IBI_CLASSPATH

   Defines the additional Java Class directories or full-path jar names which will be available for Java Services (if not included in your system CLASSPATH).

   ```
   /path/to/hive-jdbc-<version>standalone.jar
   ```

2. You also need to configure the server for inbound Kerberos.
3. Click *Configure*.

Configuring the DataMigrator or Reporting Server for Inbound Kerberos

A server started with security provider OPSYS can be configured for Kerberos connections.

To implement the single sign on Kerberos security:

1. On the Access Control page, right-click the OPSYS provider, and select *Properties* from the context menu.

   The OPSYS Security Configuration page opens.

2. In the krb5_srv_principal * field, enter your server principal used for Kerberos security.
3. Click *Save*.

   The edaserve.cfg file is updated with this attribute.
4. On the Workspace page, expand the Special Services and Listeners folder, right-click *TCP/HTTP*, and select *Properties of TCP*.
5. Check SECURITY=KERBEROS.
6. Click *Save and Restart Listener*.

   The odin.cfg file is updated with this attribute.

When the server is started, a user can connect to the Web Console from Internet Explorer without a prompt for user ID and password. The *Login Info* shows connection type *Kerberos*. The connection is done using the Kerberos ticket from the browser. The connected user ID is derived from this ticket.

Connection to the server requires that there is a local OPSYS user ID with the same name as the Kerberos user ID on the operating system running the server. This user ID is used for tscom3 process impersonation.

If a user signs off from the Kerberos connection, the user can make explicit connections with the local Unix user ID and password. Connection with another Kerberos user ID as an explicit connection will not work.

Configuring Java Authentication and Authorization Services

Once you configure a DataMigrator or WebFOCUS Reporting Server for access to Hadoop through Hive, you must also configure Java to use the HTTP service principal.

1. Using the name of the HTTP service principal for your server that you specified when configuring the DataMigrator or WebFOCUS Reporting Server for inbound Kerberos, store the credentials for this principal in a keytab. Enter the following line:

   ```
   $ klist -k /etc/krb5.keytab
   ```

   Confirm the credentials are stored correctly, as follows:

   ```
   Keytab name: FILE:/etc/krb5.keytab
   KVNO Principal
   ---- --------------------------------------
       1 HTTP/server.example.com@REALM.COM
   ```

2. Create a jaas.conf file in the location of your choice, using your preferred text editor, for example, /ibi/srv82/wfs/etc/jaas.conf. The example code also shows sample values for the keyTab and principal parameters.

```
jconnection {
 com.sun.security.auth.module.Krb5LoginModule required
 debug=false doNotPrompt=true useKeyTab=true
 keyTab="/etc/krb5.keytab"
 storeKey=true isInitiator=false
 principal="HTTP/server.example.com@REALM.COM";;
};
com.sun.security.jgss.accept {
 com.sun.security.auth.module.Krb5LoginModule required debug=false
 doNotPrompt=true useKeyTab=true
 keyTab="/etc/krb5.keytab"
 storeKey=true isInitiator=false
 principal="HTTP/server.example.com@REALM.COM";;
};
```

3. From the Web Console, click the *Workspace* folder.
4. Expand the *Java Services* folder, right-click *DEFAULT*, and click *Properties*. The Java Services Configuration window opens.
5. Expand the JVM section. In the JVM_Options box, enter the location where you saved the jaas.conf file. For example:

   ```
   -Djava.security.auth.login.config=/ibi/srv82/wfs/etc/jaas.conf
   ```

6. Click *Save and Restart Java Services*.

Troubleshooting

Here are some of the errors you may see when attempting to connect to a Hadoop cluster. These error messages will be prefixed by FOC1500.

Java listener may be down

```
ERROR: ncjInit failed.  Java listener may be down - see edaprint.log.
```

This is a Java specific issue. Confirm that your Java version meets your system requirements for bit size (32 bit vs 64 bit), and that the value for JAVA_HOME is pointing to the correct Java version.

No suitable driver found for jdbc: ...

```
(-1) [00000] JDBFOC>> makeConnection(): conn is NULLjava.sql.
SQLException: No suitable driver found for jdbc:hive1://server:10000/default
```

The name of the JDBC driver in the URL is incorrect. For example, in the above URL, the driver is *hive1*.

ClassNotFoundException

```
(-1) [00000] JDBFOC>> connectx(): java.lang.ClassNotFoundException:
org.apache.hive.jdbc.HiveDriver
```

One of the jar files that comprises the Hive JDBC driver is not found in the CLASSPATH, or the driver name is incorrect. In the above example, the driver name is org.apache.hive.jdbc.HiveDriver.

Required field 'client_protocol' is unset!

```
(-1) [00000] JDBFOC>> makeConnection(): conn is NULLjava.sql.
SQLException: Could not establish connection to
jdbc:hive2://server:10000/default;principal=hive/server@REALM.
COM;auth=kerberos;kerberosAuthType=fromSubject: Required field
'client_protocol' is unset! Struct:TOpenSessionReq(client_protocol:null,
configuration:{use:database=default})
```

This is due to a Java version mismatch. The version of Java on the client is newer than the one on the server. To resolve this error, update Java on the system where you are running our server.

Could not open client transport with JDBC Uri:

```
-1) [00000] JDBFOC>> makeConnection(): conn is NULLjava.sql.
SQLException: Could not open client transport with JDBC Uri:
jdbc:hive2://server:10000/default;principal=hive/server@REALM.
COM;auth=kerberos;kerberosAuthType=fromSubject: ...
```

This is a generic error, and could be due to any kind of error in the URL or configuration. For more details, read the rest of the message.

Unsupported mechanism type PLAIN

```
 (-1) [00000] JDBFOC>> makeConnection():  conn is NULLjava.sql.
SQLException: Could not open client transport with JDBC Uri:
jdbc:hive2://server:10000/default: Peer indicated failure: Unsupported
mechanism type PLAIN
```

The cluster has Kerberos enabled and a Kerberos principal to authenticate against is not specified. To resolve this error, add the following to the end of the URL using your cluster and REALM name:

```
;principal=hive/server@REALM.COM
```

UnknownHostException: *server*

```
(-1) [00000] JDBFOC>> makeConnection(): conn is NULLjava.sql.
SQLException: Could not open client transport with JDBC Uri:
jdbc:hive2://server:10000/default;principal=hive/server@REALM.
COM;auth=kerberos;kerberosAuthType=fromSubject: java.net.
UnknownHostException: name
```

The server you specified could not be reached. Either the system is down or the name is invalid.

Could not open client transport with JDBC Uri:... Connection refused

```
(-1) [00000] JDBFOC>> makeConnection(): conn is NULLjava.sql.
SQLException: Could not open client transport with JDBC Uri:
jdbc:hive2://server:10000/default;principal=hive/server@REALM.
COM;auth=kerberos;kerberosAuthType=fromSubject: java.net.
ConnectException: Connection refused: connect
```

The hostname could be reached, but there is no Hive Thrift server running at the hostname and port number you specified

Client not found in Kerberos database

```
(-1) [00000] JDBFOC>> makeConnection(): javax.security.auth.login.
LoginException: Client not found in Kerberos database (6)
```

The user ID (User Principal Name) specified is not a valid Kerberos user ID.

Pre-authentication information was invalid

```
-1) [00000] JDBFOC>> makeConnection():
javax.security.auth.login.LoginException: Pre-authentication
information was invalid (24)
```

The user ID (User Principal Name) specified was found, but the password is missing or invalid.

GSS initiate failed

```
(-1) [00000] JDBFOC>> makeConnection(): conn is NULLjava.sql.
SQLException: Could not open client transport with JDBC Uri:
jdbc:hive2://server:10000/default;principal=hive/server@domain.
COM; GSS initiate failed
Caused by: org.apache.thrift.transport.TTransportException:
```

Kerberos was unable to authenticate against the specified Service Principal Name.

This occurs if you are using static credentials that were obtained by kinit before the server starts, and did not specify the option to allow that. To resolve this issue, from the Web Console:

1. Click *Workspace* in the side bar.
2. Expand the *Java Services* folder.
3. Right-click *DEFAULT*.
4. Click *Properties*.
5. Expand the *JVM Settings* section.
6. Enter the following in the JVM_OPTIONS box:

   ```
   -Djava.security.auth.useSubjectCredsOnly=false
   ```

This also occurs if you are using subject credentials with explicit or password pass through security, but did not specify that you are doing so. Add the following to the end of the URL:

```
;auth=kerberos;kerberosAuthType=fromSubject
```

Could not create secure connection to ... Failed to open client transport

```
(0) [ 08S0] Could not create secure connection to
jdbc:hive2://server:10000/default;principal=hive/server@REALM.
COM;auth=kerberos;kerberosAuthType=fromSubject: Failed to open client
transportjava.sql.SQLException: Could not create secure connection to
jdbc:hive2://server:10000/default;principal=hive/server@REALM.
COM;auth=kerberos;kerberosAuthType=fromSubject: Failed to open client
transport
```

The client was not able to use Kerberos to connect. You did not enable Kerberos and are using subject credentials. To resolve this issue, add the following to the server profile:

```
ENGINE SQLHIV SET ENABLE_KERBEROS ON
```

Cannot locate default realm

```
(-1) [00000] JDBFOC>> makeConnection(): javax.security.auth.login.
 LoginException: KrbException: Cannot locate default realm
```

Kerberos looks for the default realm in /etc/krb5.conf, or on Windows krb5.ini, and it either cannot find or read the file or the default realm is not specified.

For a server running as a service on a Windows machine, you must explicitly specify the realm for the organization and the KDC. Review the file to make sure that the default realm is specified and correct. To resolve this issue, from the Web Console:

1. Click *Workspace* in the side bar.
2. Expand the *Java Services* folder.
3. Right-click *DEFAULT*.
4. Click *Properties*.
5. Expand the *JVM Settings* section.
6. Using the name of your realm and Key Distribution Center (KDC), enter the following in the JVM_OPTIONS box:

   ```
   -Djava.security.krb5.realm=REALM.COM
   -Djava.security.krb5.kdc=kdc.domain.com
   ```

Click *Save and Restart Java Services*.

Creating Synonyms With Apache Hive

Synonyms define unique names, or aliases, for each Hive table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Hive table.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.

   The Applications page opens.

2. On the Configured list, right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.

4. After entering the parameter values, click the *Add* button.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Using Direct Pass-through With Apache Hive

Direct Pass-through commands, such as SHOW TABLES and DESCRIBE tablename, do not display any results.

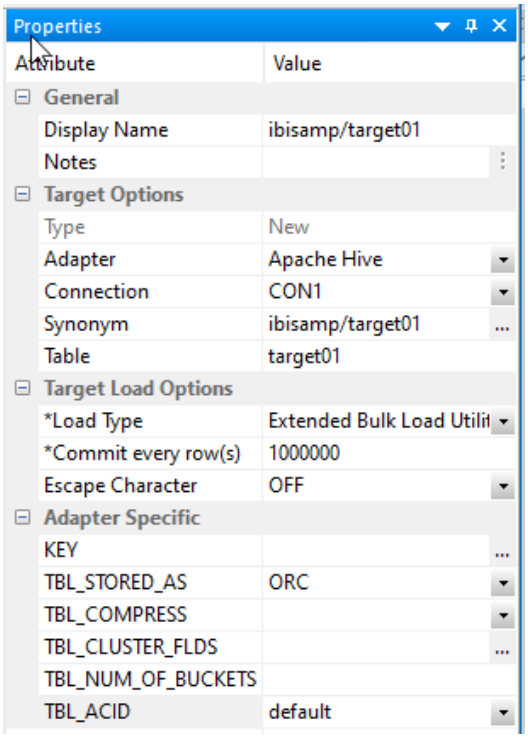
This can be avoided by adding a comment:

```
ENGINE SQLHIV
/*QUERY*/ SHOW TABLES ;
END
```

Loading Data Using DataMigrator

DataMigrator can be used to load data from any accessible data source into Hadoop and create metadata in Hive. Currently, the only load type supported is Extended Bulk Load. The exception are files stored as ORC (Optimized Row Columnar) format, for which Insert/Update can be used.

The table should be stored as TEXTFILE (default), ORC, or Parquet, as shown in the following image.



For a new ORC table, you can use the TBL_ACID property to specify whether transactions are supported. The default is the setting specified in Hive. You can select Yes, if you want the new Hive table to support transactions, or No, if you do not want it to support new transactions.

You can set the Escape Character property to ON to properly process data with non-printable characters.

For more information, see the *DataMigrator User's Guide*.

Chapter 10 Using the Adapter for Apache Phoenix to Access HBase

Apache Phoenix is a query engine for Apache HBase and MapR-DB.

In this chapter:

Preparing the Apache Phoenix Environment

The following components are needed to use the adapter for Apache Phoenix:

- **Java.** To use this Java-based adapter, you must have Java, Version 1.8 or later, installed. Java can be downloaded from *http://www.java.com*.

  The location of the Java must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/lib/jvm/jre-1.8.0
  ```

  If you have JDK installed:

  ```
  export JAVA_HOME=/usr/lib/jvm/jdk-1.8.0
  ```

  If you are using Windows, right-click *Computer* and select *Properties*. Then click *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

  ```
  C:\Program Files\Java\jdk-8\bin\server;C:\Program Files\Java\jdk-8\bin;
  ```

- **JDBC Driver.** The JDBC driver consists of a jar file distributed with Apache Phoenix. If you are installing the server on the same system where Apache Phoenix is installed, you can point to the jar file as described in the next section. If you are installing the server on some other system, copy the file to a location of your choice.

Procedure: How to Configure the Java CLASSPATH

The location of the Apache Phoenix JDBC Driver must be specified to the server. If you are running the server on the same system as a Drillbit, you can specify its location. If you are running the server on another system, copy the files listed below to some location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console menu bar, select *Workspace*

   or

   From the Data Management Console, expand the *Workspace* folder.

2. Expand the *Java Services* folder. Right-click *DEFAULT* and click *Properties*.

   The Java Services Configuration page opens.

3. Click on the chevrons to expand Class path.

   In the IBI_CLASSPATH box, enter the full location of the jar file. For example:

   ```
   /usr/lib/phoenix/phoenix-client.jar
   ```

   Note:

   - The file names must be entered one per line.

     If you are installing the adapter on a different system than where you have Apache Phoenix installed, copy the jar file to a location on that system.

   - For a server running on Windows, use Windows syntax for directory names. For example:

     ```
     C:\phoenix\phoenix-client.jar
     ```

   - If your Hadoop cluster is Kerberos enabled, add the name of the directory containing the configuration files hbase-site.xml and core-site.xml. By default, if the server is installed on the same system where Hadoop or HBase were installed, these files are located in the /etc/hbase/conf directory. If the server is installed on a different system, use the name of the directory where you copied the configuration files. For example:

     - **For a Linux system:**

       ```
       /etc/hbase/conf
       ```

- **For a Windows system:**

  ```
  c:\phoenix
  ```

4. Scroll down and click the *Save and Restart Java Services* button.

Configuring the Adapter for Apache Phoenix

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Apache Phoenix Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click *Apache Phoenix* and click *Configure*.
5. In the URL box, enter the URL used to connect to your Drillbit. For more information, see *Apache Phoenix URL Configuration Settings* on page 312.
6. In the Driver Name box, enter the following:

   ```
   org.apache.phoenix.jdbc.PhoenixDriver
   ```

7. For Security, select *Trusted*.

   Note: Explicit should not be used because Phoenix ignores any user ID that is entered; it connects with the user ID that was used to start the server. Individual user credentials can only be used with a secured (Kerberos enabled) server and are specified in the URL.

8. Click *Configure*.

Reference: Apache Phoenix URL Configuration Settings

The Adapter for Apache Phoenix is under the SQL group folder.

The URL configuration for the Apache Phoenix adapter is:

```
jdbc:phoenix:quorum:[port:[/directory[:principal[:keytab]]]]
```

where:

quorum

Is a comma separated list of ZooKeeper servers.

port

Is the port number for ZooKeeper. The default value is 2181. On a MapR cluster, the default value is 5181.

directory

Is the name of the Phoenix directory within ZooKeeper.

For a server that does not have Kerberos enabled, the default is */hbase-unsecure* .

For a Kerberos enabled server, the default is */hbase-secure*.

Note: The directory names listed above must be entered exactly as written.

principal

The Kerberos user principal. For example:

user@REALM

keytab

The Kerberos keytab containing the specified user. For example:

```
/home/user/user.keytab
```

The following is an example of an unsecured server with one zookeeper node:

```
jdbc:phoenix:clu0:2181:/hbase-unsecure
```

The following is an example for a secured server with three zookeeper nodes:

```
jdbc:phoenix:clu1,clu2,clu3:2181:/user@REALM.COM:
/user/home/user.keytab
```

Troubleshooting

If the server is unable to configure the connection, a dialog box opens with an error message. The message typically begins with the following:

```
(FOC1400) SQLCODE IS -1 (HEX: FFFFFFFF) XOPEN: nnnn
```

where:

nnnn

Is the message number returned.

Some of the more common errors are:

```
(FOC1500) : (-1) [00000] JDBFOC>> connectx():
(FOC1500) : java.lang.UnsupportedClassVersionError:
org/apache/drill/jdbc/Driver :
(FOC1500) : Unsupported major.minor version 51.0
(FOC1479) ERROR CONNECTING TO SQL DATABASE
```

This indicates that your Java version is less than 1.8, which is required.

Creating Synonyms With Apache Phoenix

Synonyms define unique names, or aliases, for each Phoenix table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Phoenix table.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

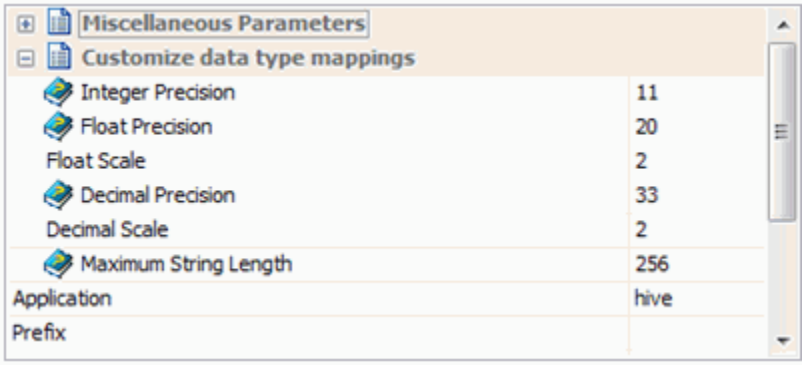
   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.

- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.
5. Optionally, expand the *Customize data type mappings* section and enter a value for Maximum String Length or numeric precision, as shown in the following image.



6. Select the check box for the objects that you want to create synonyms for. If you want to change the name of the synonym from the default name, click the name and edit it as needed.
7. Click *Create Synonym*.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Chapter 11

Using the Adapter for Apache Spark

The Adapter for Spark provides for analysis of both structured and complex data. Apache Spark is a fast and general engine for large-scale data processing.

In this chapter:

Introducing the Adapter for Apache Spark

Apache Spark is a fast and general engine for large-scale data processing

Apache Spark can run on Hadoop or in a standalone cluster mode. Data can be stored in HDFS, the local file system, or on Cloud.

The Adapter for Apache Spark uses JDBC to connect to the Spark SQL Thrift Server.

The Adapter for Apache Spark is JDBC based. The DataMigrator or WebFOCUS server can be run on any platform (including Windows) that connects to the server where Hive is running.

Preparing the Apache Spark Environment

The following components are needed to use the adapter for Hadoop/Hive:

- **Java.** To use this JDBC-based adapter, you must have Java installed. Hadoop requires Version 1.7 or later. Java can be downloaded from *http://www.java.com*.

  The location of Java must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/lib/jvm/jre-1.7.0
  ```

If you have JDK installed:

```
export JAVA_HOME=/usr/lib/jvm/jdk-1.7.0
```

If you are using Windows, right-click *Computer* and click *Properties*. Then click *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

```
C:\Program Files\Java\jdk7\bin\server;C:\Program Files\Java\jdk7\bin;
```

- **JDBC Drivers.** The Spark JDBC driver requires two jar files, which are included with Apache Spark. If you are installing the server on the same system where Spark is installed, you can point to the jar files as described in the next section. If you are installing the server on some other system, copy those files to a location of your choice.

Procedure: How to Configure the Java CLASSPATH

The location of the Apache Spark jar files must be specified to the server. If you are running the server on the same system as the Hadoop and Hive server, you can specify their location. If you are running the server on another system, copy the files listed below to some location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console menu bar, select *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.

2. Expand the *Java Services* folder. Right-click *DEFAULT* and click *Properties*.

   The Java Services Configuration page opens.

3. Expand *Class path*.

   In the IBI_CLASSPATH box, enter the full location of the Spark files shown below. The file names must be entered one per line.

   If you are installing the adapter on a different system than where Hadoop and Hive are installed, copy the jar files to a location on that system.

   Enter the location on your system that matches your installation.

   ```
   hive-jdbc-<version>-standalone.jar
   hadoop-common.jar
   ```

Configuring the Adapter for Apache Spark

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Adapter for Apache Spark

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click *Apache Spark SQL* and click *Configure*.

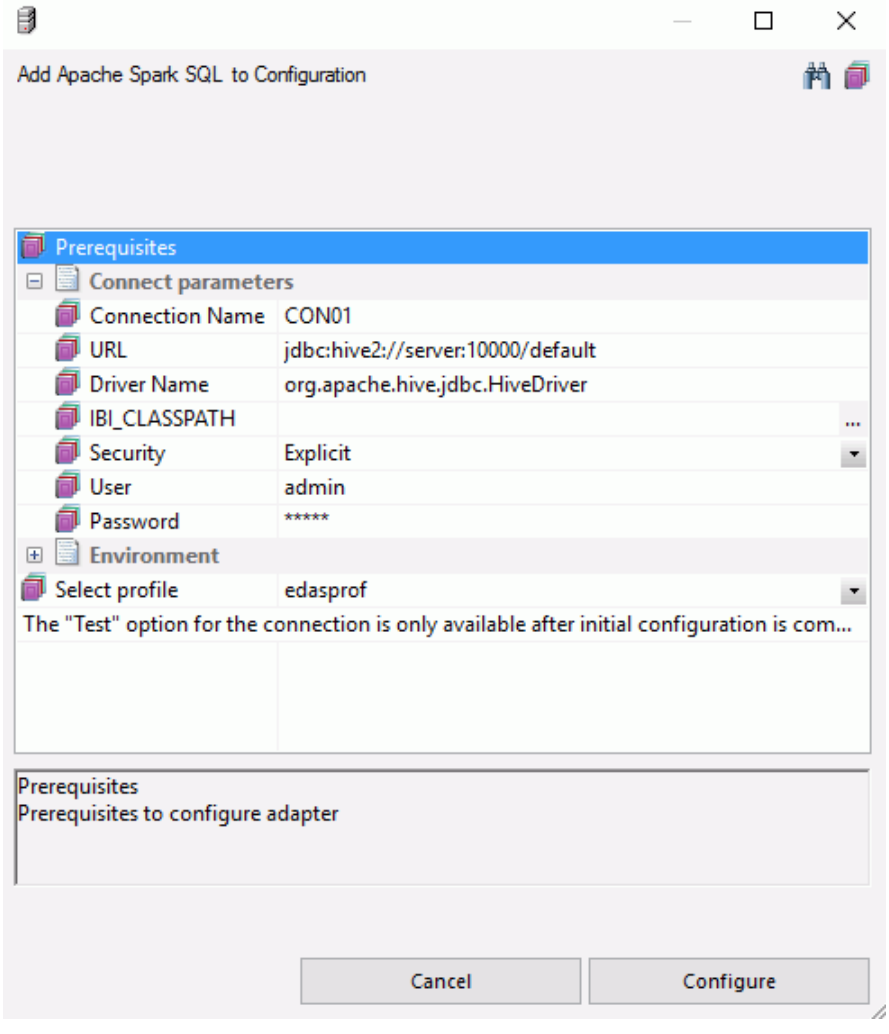
5. In the URL box, type the URL used to connect to your Hive server. For more information, see *Hive/Impala Adapter Configuration Settings*.

6. In the Driver Name box, type the driver name from the following table:

Driver	JDBC Driver Name
Apache Spark	`org.apache.hive.jdbc.HiveDriver`

7. Select the security type. If you are using Explicit, type your user ID and password.

The following image shows an example of the configuration settings used:



8. Select *edasprof* from the Select profile drop-down menu to add this connection for all users, or select a user profile.
9. Click *Configure*.
10. Click *Test*. You should see a list of data sources on your server.

Reference: Adapter for Apache Spark Configuration Settings

The Adapter for Hadoop/Hive is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the location of the data source.

The URL used depends on what type of server you are connecting to. See the table below for examples.

Server	URL
Hiveserver2	`jdbc:hive2://server:10000/default`
Kerberos Hiveserver2 (static)	`jdbc:hive2://server:10000/default; principal=hive/server@REALM.COM`
Kerberos Hiveserver2 (user)	`jdbc:hive2://server:10000/default;principal=hive/server@REALM.COM; auth=kerberos;kerberosAuthType=fromSubject`

where:

server

Is the DNS name or IP address of the system where the Hive server is running. If it is on the same system, localhost can be used.

default

Is the name of the default database to connect to.

10000

Is the default port number Spark Thrift Server is listening on if not specified when the Hive server is started.

REALM.COM

For a Kerberos enabled Hive server, this Is the name of your realm.

Driver Name

Is the name of the JDBC driver, org.apache.hive.jdbc.HiveDriver.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. This value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions when setting values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar). When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Kerberos

Connections to a Hive server with Kerberos enabled can be run in one of two ways:

- **Static.** The same Kerberos credential is used for all connections. You must obtain a Kerberos ticket before starting the server. On expiration of the Kerberos ticket, a new one must be obtained and the server must be restarted.

  This mode is useful for testing, but is not recommended for production deployment.

- **User.** Each user connecting to the server connects to Hive using their own credentials from the Adapter for Apache Spark connection in the user profile. The server obtains a Kerberos ticket for the user.

To setup connections to a Kerberos enabled Spark Thrift Server instance if each user has their own connection, the Reporting Server has to be secured. The server can be configured with security providers PTH, LDAP, DBMS, OPSYS, or Custom, as well as multiple security provider environments.

Kerberos Static Ticket Requirements

In this configuration, all connections to the Spark Thrift Server instance will be done with the same Kerberos user ID derived from the Kerberos ticket that is created before the server starts.

1. Create Kerberos ticket using:

   ```
   kinit kerbid01
   ```

   where:

   kerbid01

   Is a Kerberos ID.

2. Verify Kerberos ticket using *klist*. The following message should be returned:

   ```
   Ticket cache: FILE:/tmp/krb5cc_532
   Default principal: kerbid01@REALM.COM

   Valid starting     Expires            Service principal
   04/29/16 16:26:50  04/30/16 02:26:53  krbtgt/REALM.COM@REALM.COM
       renew until 05/06/16 16:26:50
   ```

3. Before configuring the Spark Thrift Server connection to a Kerberos-enabled instance, the connection should be tested. Use Beeline, the native tool, to test it.

4. Start the server in the same Linux session where the Kerberos ticket was created. Log in to the Web Console and click the *Adapters* tab.

5. Right-click *Apache Spark SQL*. Use the following parameters to configure the adapter:

   URL

   Enter the same URL that you used to connect to the Hive Instance using Beeline.

   ```
   jdbc:hive2://server:10000/default;principal=hive/server@REALM.COM
   ```

   Security

   Set to *Trusted*.

6. In the Select profile drop-down menu, select the *edasprof* server profile.
7. Click *Configure*.
8. Next, configure Java services. Click the *Workspace* tab and expand the *Java Services* folder.
9. Right-click *DEFAULT* and click *Properties*.
10. Expand the *JVM Settings* section. In the JVM options box, add the following:

    ```
    -Djavax.security.auth.useSubjectCredsOnly=false
    ```

11. Restart Java services.

Once these steps are completed, the adapter can be used to access a Kerberos-enabled Spark Thrift Server instance.

Kerberos User Ticket Requirements

In this configuration, each connected user has a Hive Adapter connection with Kerberos credentials in the user profile.

1. Enable multi-user connection processing for Kerberos by adding the following line to your profile (edasprof.prf):

   ```
   ENGINE SQLSPK SET ENABLE_KERBEROS ON
   ```

2. Configure the Hive Adapter Connection in the user profile using the following values:

URL

Is the URL to the location of the data source.

Server	URL
Kerberos Hiveserver2 (User)	`jdbc:hive2://server:10000/default;principal=hive/server@REALM.COM; auth=kerberos;kerberosAuthType=fromSubject`

Security

Set to Explicit.

User and Password

Enter your Kerberos user ID and password. The server will use those credentials to create a Kerberos ticket and connect to a Kerberos-enabled Spark Thrift Server instance.

Note: The user ID that you use to connect to the server does not have to be the same as the Kerberos ID you use to connect to a Kerberos-enabled Spark Thrift Server instance.

Select Profile

Select your profile or enter a new profile name consisting of the security provider, an underscore and the user ID. For example, *ldap01_pgmxxx*.

3. Click *Configure*.

Troubleshooting

If the server is unable to configure the connection, an error message is displayed. An example of the first line in the error message is shown below, where *nnnn* is the message number returned.

```
(FOC1400) SQLCODE IS -1 (HEX: FFFFFFFF) XOPEN: nnnn
```

Some common errors messages are:

```
[00000] JDBFOC>> connectx(): java.lang.UnsupportedClassVersionErr : or: org/
apache/hive/jdbc/HiveDriver : Unsupported major.minor version 51.0
```

The adapter requires Java 1.7 or later and your JAVA_HOME points to Java 1.6.

```
(FOC1500)  :  ERROR: ncjInit failed. failed to connect to java server: JSS
(FOC1500)  :   . JSCOM3 listener may be down - see edaprint.log for details
```

The server could not find Java. To see where it was looking, review the edaprint.log and set JAVA_HOME to the actual location. Finally, stop and restart your server.

```
(FOC1260)  (-1) [00000] JDBFOC>> connectx():
java.lang.ClassNotFoundException:
   org.apache.hive.jdbc.HiveDriver
(FOC1260) Check for correct JDBC driver name and environment variables.
(FOC1260) JDBC driver name is org.apache.hive.jdbc.HiveDriver
(FOC1263) THE CURRENT ENVIRONMENT VARIABLES FOR SUFFIX SQLHIV ARE :
(FOC1260) IBI_CLASSPATH : ...
```

The JDBC driver name specified cannot be found in the jar files specified in IBI_CLASSPATH or CLASSPATH. The names of the jar files are either not specified, or if specified, do not exist in that location.

```
[08S0] Could not establish connection to jdbc:hive2://hostname:10000:
java.net.UnknownHostException: hostname
```

The server hostname could not be reached on the network. Check that the name is spelled correctly and that the system is running. Check that you can ping the server.

```
[08S01] Could not establish connection to localhost:10000/default:
: java.net.ConnectException: Connection refused
```

The Spark Thrift server is not listening on the specified port. Start the server if it is not running, and check that the port number is correct.

Creating Synonyms With Apache Spark

Synonyms define unique names, or aliases, for each Hive table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Hive table.

Procedure: How to Create a Synonym With Apache Spark

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.

2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Using Direct Pass-through With Apache Spark

Direct Pass-through commands, such as SHOW TABLES and DESCRIBE tablename, do not display any results.

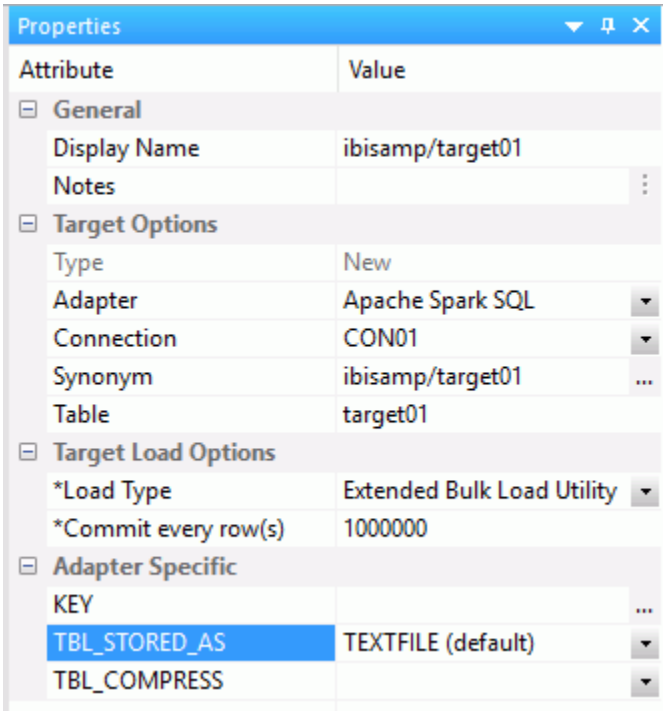
This can be avoided by adding a comment:

```
ENGINE SQLSPK
/*QUERY*/ SHOW TABLES ;
END
```

Loading Data into Apache Spark Using DataMigrator

DataMigrator can be used to load data from any accessible data source into Hadoop and create metadata in Hive. Currently, the only load type supported is Extended Bulk Load. The exception are files stored as ORC (Optimized Row Columnar) format, for which Insert/Update can be used.

The table should be stored as TEXTFILE (default), ORC, or Parquet, as shown in the following image.



For more information, see the *DataMigrator User's Guide*.

Chapter 12

Using the Adapter for Axiom EPM

The Adapter for Axiom EPM provides data access for reporting from the following Axiom EPM Table Types: Data, Reference, and DocumentBasedReferenced. The adapter fully enforces Axiom EPM security.

The adapter provides three types of services:

- Authentication services.
- Axiom EPM data source connection management services.
- Metadata administration and data access security services.

In this chapter:

Preparing the Axiom EPM Environment

Prior to configuring the Adapter for Axiom EPM, you must ensure that minimal software requirements have been met. In addition, you must define an Axiom EPM Access ID to enforce Axiom EPM security. Administrators must also review available connection options before beginning the initial configuration.

Software Requirements

The following software is required on the server to support the Adapter for Axiom EPM:

- An appropriate RDBMS connectivity product on the computer where you will install the Adapter for Axiom EPM server. This is to provide access to an Axiom EPM data source server for operational data access.

Note: For information about Axiom EPM with versions of software other than those listed in this topic, contact your Information Builders representative.

Setting Up the Axiom EPM Access ID

The Adapter for Axiom EPM initially authenticates with Axiom EPM using an Axiom EPM Administrator user ID and password. If successful, the adapter attaches to the RDBMS using an access ID of your choice. Typically, this access ID is an RDBMS ID dedicated to the adapter. If required, this access ID can also be an Axiom Administrator ID.

In addition, the access ID must have read access to any Axiom EPM Tables that you wish to report against.

Note: The Axiom Adapter is not supported when the Server is running in Pooled deployment mode.

Configuring the Adapter for Axiom EPM

Before configuring the adapter for Axiom EPM, you must first register the Axiom EPM application (AXIOMCAT) in the catalog path, add the underlying SQL adapter, and add the REST adapter for administration and security information retrieval.

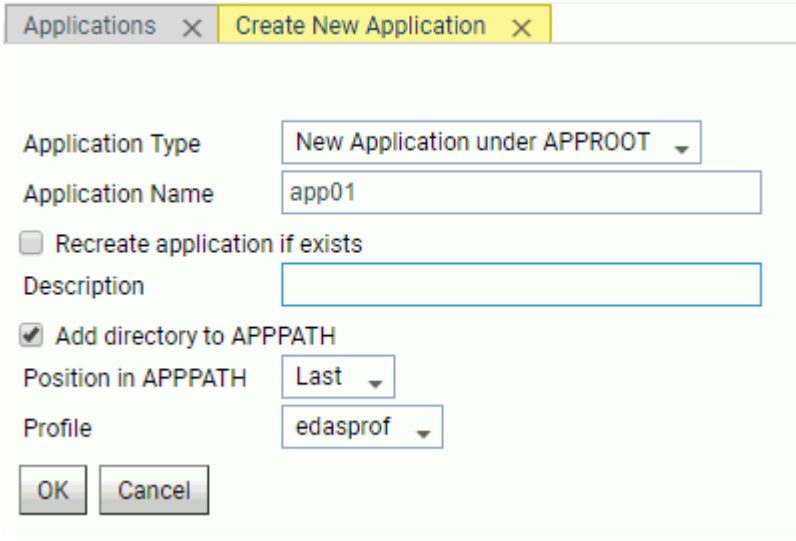
Creating the AXIOMCAT Application

Before you can configure the Axiom EPM adapter, the AXIOMCAT application must be created and added to the application path of the server. This can be done through the Web Console.

Procedure: How to Create the AXIOMCAT Application

1. From the Web Console menu bar, select *Applications*.
2. Right-click the *Application Directories* folder and select *New*, *Application Directory*.

The Create New Application window opens, as shown in the following image.



3. Change the Application Name to *axiomcat*. Make sure it is all in lowercase.
4. Click *OK*.

Adding the SQL Database Adapter

The Adapter for Axiom EPM is an enhanced version of the underlying data source adapter. The adapter must be added into the server configuration.

When you add an SQL adapter, there is no need to provide configuration parameters. Instead, these configuration parameters are supplied later when configuring connections to Axiom EPM data sources.

Example: Adding an Adapter

The Web Console provides graphical point and click tools to add and test the adapter.

1. From Web Console sidebar, click *Connect to Data*.
2. Expand the *Available* folder, if it is not already expanded.
3. Expand the *SQL* group folder and a *MS SQL Server* or *Oracle* folder.
4. Right-click a version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. For Oracle, no configuration parameters are required. For MS SQL Server, supply any value for the mandatory *Server* parameter. It does not have to be a valid MS SQL server name. *Myserver* is adequate.
6. Click *Configure*.

   The adapter is added to your server configuration.

Adding the REST Adapter and Connection

The Adapter for Axiom EPM uses REST API calls to obtain metadata and security information from the Axiom EPM environment. To do this, a connection to Axiom needs to be configured.

Procedure: How to Configure the REST Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

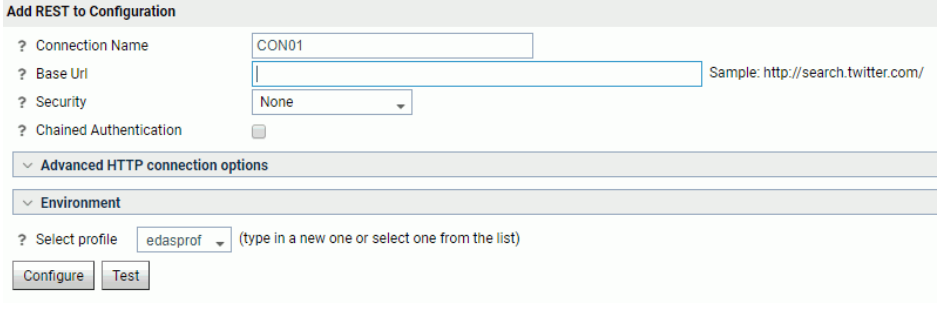
1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click *REST*, and select *Configure*.

   The Add REST to configuration pane opens, as shown in the following image.



5. Provide the following information:

   Connection Name

   Enter the following value exactly as shown:

   AxiomToken

   Service Description

   None

   Base URL

   Enter the URL of the machine where the Axiom software is installed in the following format:

   `http://machine.name.com`

   Security

   Select *Password Passthru* from the drop-down menu.

6. Select the *Chained Authentication* check box.

   Provide the following information:

   Additional Chained Authentication Information

   Authentication synonym

   Enter the following value in lowercase:

   `axgettok`

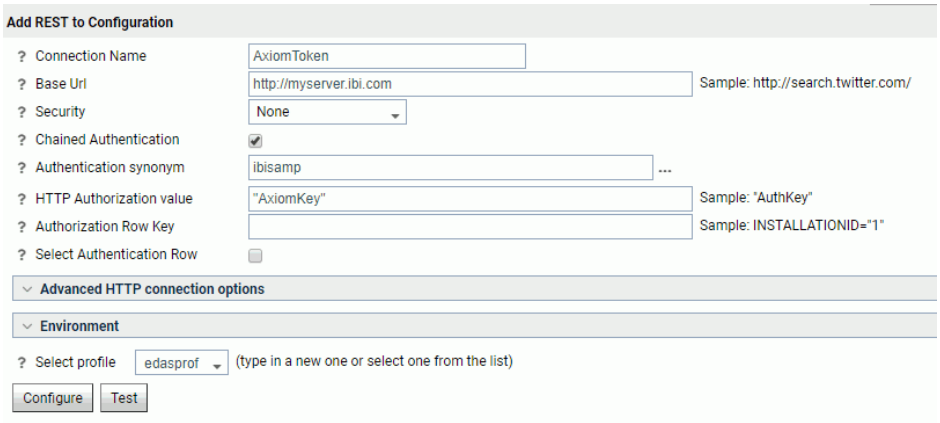
   HTTP Authorization value

   Enter the following (including double quotation marks):

   `"AxiomKey"`

7. Make sure the *Select profile* drop-down menu points to *edasprof*.

The pane should now look like the following image, except for the Base URL, which will reflect an installation value.



8. Click *Configure*.

Adding the First Connection to Axiom EPM

Although multiple Axiom EPM connections can be created, the first connection requires slightly different management steps. The reason for this is that before any connections to Axiom EPM data sources can occur, a base layer of application metadata is created and configured. This occurs transparently for the Administrator, but is a slightly different connection management process than when adding additional connections.

To create the first connection:

- ❑ Select an existing Axiom Administrator ID.

  The Administrator ID is the value that is used as a metadata access key. To access the Web Console, the administrator must know this ID. Users outside of the Web Console do not have metadata access without first logging in to Axiom EPM through this integration.

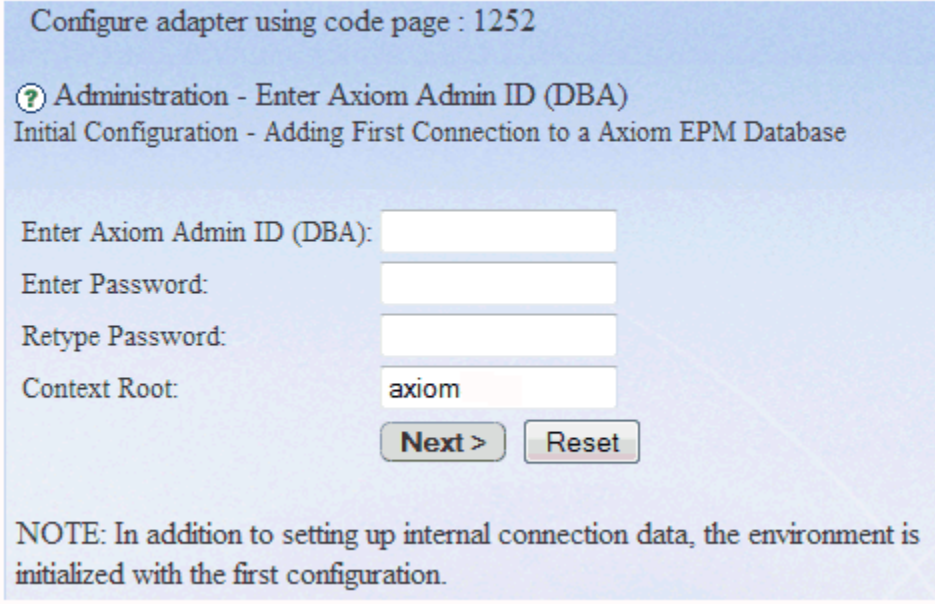
- ❑ Add an Axiom EPM Connection.

Procedure: How to Add the First Axiom EPM Connection

1. Open the Web Console.
2. From Web Console sidebar, click *Connect to Data*.
3. Expand the *Available* folder, if it is not already expanded.
4. Expand the *ERP* group folder.

5. Right-click *Axiom EPM*, and select *Configure*.
6. Click *Configure*.
7. Click *Next*.

   The Administration - Enter Axiom Admin ID (DBA) opens, as shown in the following image.



8. In the *Enter Axiom Admin ID (DBA)* field, enter the administrator ID that has access to all Axiom instances (and Tables) that you what to configure the adapter for.
9. Enter and retype the password for the administrator ID in the corresponding fields.
10. Accept the default in the *Context Root* field or enter a different one. This is the context root for the Axiom API. It is case sensitive and must match the value entered during the installation and configuration of the Axiom product. The default value is axiom.

    Note: This field is only shown when the adapter is first configured.

11. Click *Next*

    The Administration - Create a New Axiom EPM Connection window opens.

12. Type the necessary information, test the connection, and if successful, click *Configure*. If not successful, edit the information until the test is successful, then click *Configure*. For more information, see step 4 in *How to Add an Axiom EPM Data Source Instance* on page 346.

    Note: The *Configure* button becomes available once the connection test to the selected RDBMS is successful.

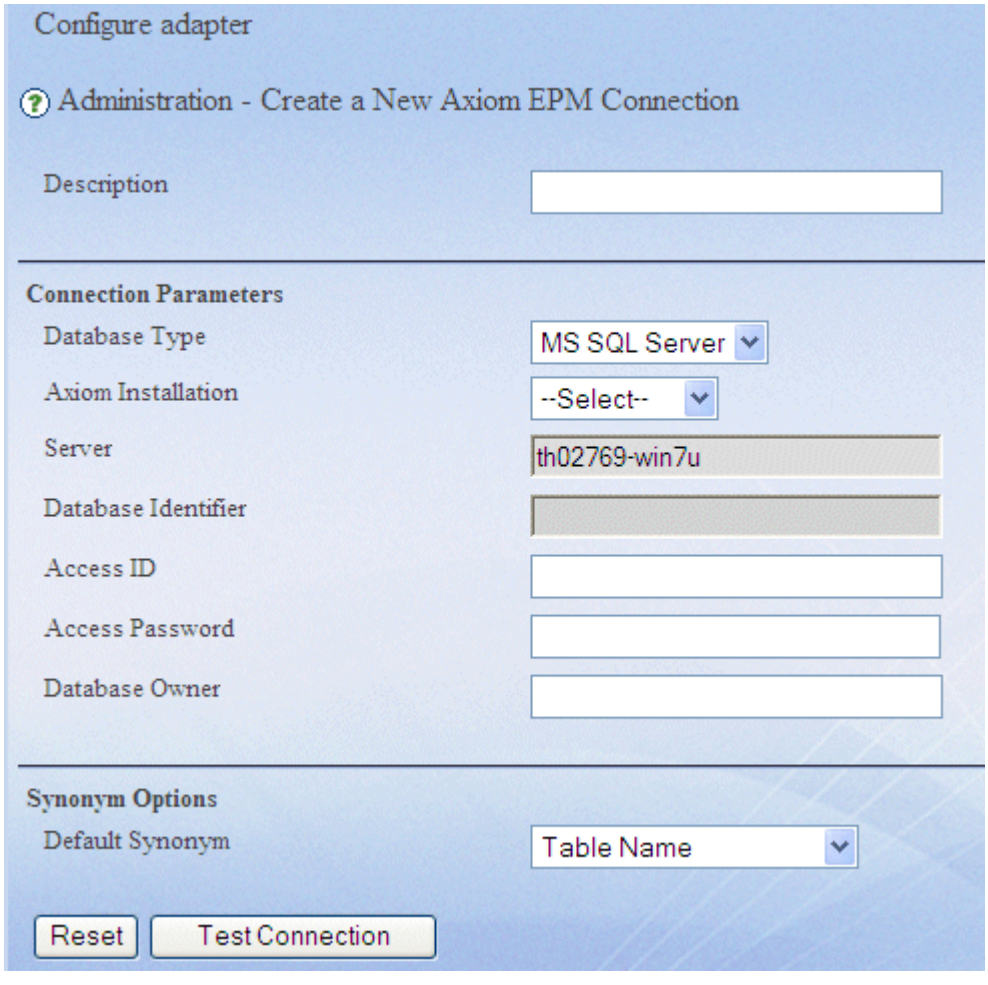
13. Once the configuration finishes, click *Next*.

    Most of the parameters can be changed in the future, but the connection cannot be created without initially supplying correct data source connectivity information. For details on the required information, see *Administration - Create a New Axiom EPM Connection Window* on page 335.

    Tip: A worksheet is provided to assist you in gathering the information you will need to type on this screen. For details, see *Axiom EPM Connection Worksheet* on page 337.

    Once completed, you may begin managing your new Axiom EPM Connection by adding synonyms.

Reference: **Administration - Create a New Axiom EPM Connection Window**



The Administration - Create a New Axiom EPM Connection window contains the following fields/options:

`Description`

Is a description that identifies the current server configuration.

Connection Parameters

`Database Type`

Is the Data source type. The options are: Oracle and MS SQL Server.

`Axiom Installation`

This drop-down box lists available Axiom installations. Select the installation that you will configure. Selecting this value will automatically set the values for the following two options (Server and Database Identifier).

`Server`

Name of the Database Server (for SQL Server only, not Oracle). This is automatically set by your selection for Axiom Installation.

`Database Identifier`

Is the data source identifier or system data source name. This is automatically set by your selection for Axiom Installation.

`Access ID`

Is the RDBMS login ID for data source connectivity. Typically, it is the same login ID Axiom EPM uses to access the data source. There are minimum data access requirements that the ID must have. For details, see *Setting Up the Axiom EPM Access ID* on page 328.

`Access Password`

Is a password associated with the access ID.

`Database Owner`

Is the data source owner identifier or the schema owner depending on your data source. The owner ID is used to fully qualify SQL requests passed to the data source. This allows multiple data source instances or schemas to exist in a single data source. Typical defaults are SYSADM on Oracle and dbo on Microsoft SQL Server.

Synonym Options

`Default Synonym`

Is the name of the synonym to default to when creating new metadata. The options are: Table Name, Table Name with Prefix, Table Name with Suffix.

Note: If you specify any of the options with Prefix or Suffix, the following additional input box is displayed.

`Prefix/Suffix Default`

Enter the prefix or suffix you wish to prepend or append to the name of the synonym. The maximum length for a Prefix/Suffix is nine characters.

Reference: **Axiom EPM Connection Worksheet**

Use the following worksheet to record connection configuration information and keep it for future reference.

Option Name	Response	Description
Description		Row.
Administrator (DBA)		Enter the administrator ID and password that has access to all instances and Tables that are required to be administered by the adapter. Must be consistent across multiple connections.
Database Type		Designation for RDBMS. The options are Oracle and MS SQL Server.
Axiom Installation		Is the installation that you will configure. Selecting this value from the drop-down list will automatically set the values for the following two options (Server and Database Identifier). ❏ Server is the name of the Database Server (for SQL Server only, not Oracle). This is automatically set by your selection for Axiom Installation. ❏ Database Identifier is the data source identifier or system data source name. This is automatically set by your selection for Axiom Installation.
Access ID		ID used by Axiom EPM to connect to the RDBMS.
Access Password		Password associated with the access ID.
Database Owner		Data source owner ID for the Axiom EPM data source.

Option Name	Response	Description
Default Synonym		The name of the synonym to default to when creating new metadata. ❏ Table Name ❏ Table Name with Prefix ❏ Table Name with Suffix **Note:** If you specify any of the options with Prefix or Suffix, an additional input box is displayed for your Prefix or Suffix entry.
Prefix/Suffix Default		Enter the prefix or suffix you wish to prepend or append to the synonym name. The maximum length is nine characters.

Managing Axiom EPM Metadata

The Adapter for Axiom EPM enables management of Axiom EPM metadata. Processes exist that manage the accessibility of Axiom EPM Tables by maintaining the metadata catalog on the reporting server. Before reports can be created or run, this metadata catalog must be created.

You can perform the following tasks to accomplish this:

- ❏ Create synonyms for Axiom EPM Tables.
- ❏ Remove synonyms.
- ❏ Update synonyms.

Accessing the Adapter for Axiom EPM Administrator

To access the environment for administration, you must have access to the Web Console and knowledge of the Axiom administrator ID and password (DBA).

Procedure: How to Access the Adapter for Axiom EPM Administration

1. Open the Web Console.
2. From Web Console sidebar, click *Connect to Data*.
3. In the navigation pane, right-click *Axiom EPM* from the list of configured adapters, and click *Properties*.

4. Type the administrators ID and password, and click *Next*.

   The Axiom EPM menu opens.

Creating Synonyms

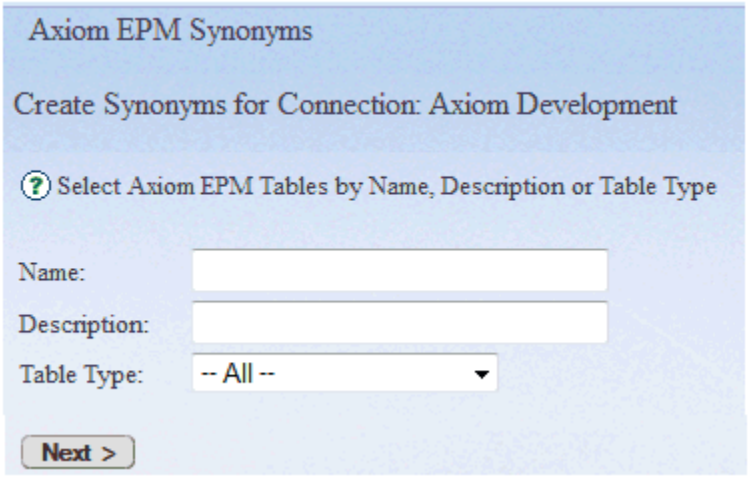
Creating synonyms for Axiom EPM is one of the core functions of the Axiom EPM Administrator. The adapter enables the administrator to choose which Axiom EPM tables are defined in the metadata catalog.

You can create synonyms using a filtered search. A search template option is provided to speed the process of searching for specific Axiom EPM Tables. This is a useful method of locating tables by name, description, or table type.

Procedure: How to Create Synonyms Using a Filtered Search

1. In the Axiom EPM menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the Axiom EPM menu, see *Accessing the Adapter for Axiom EPM Administrator* on page 338.
2. Click *Create* under the *Synonyms* group.

The Create Synonyms for Connection window opens.



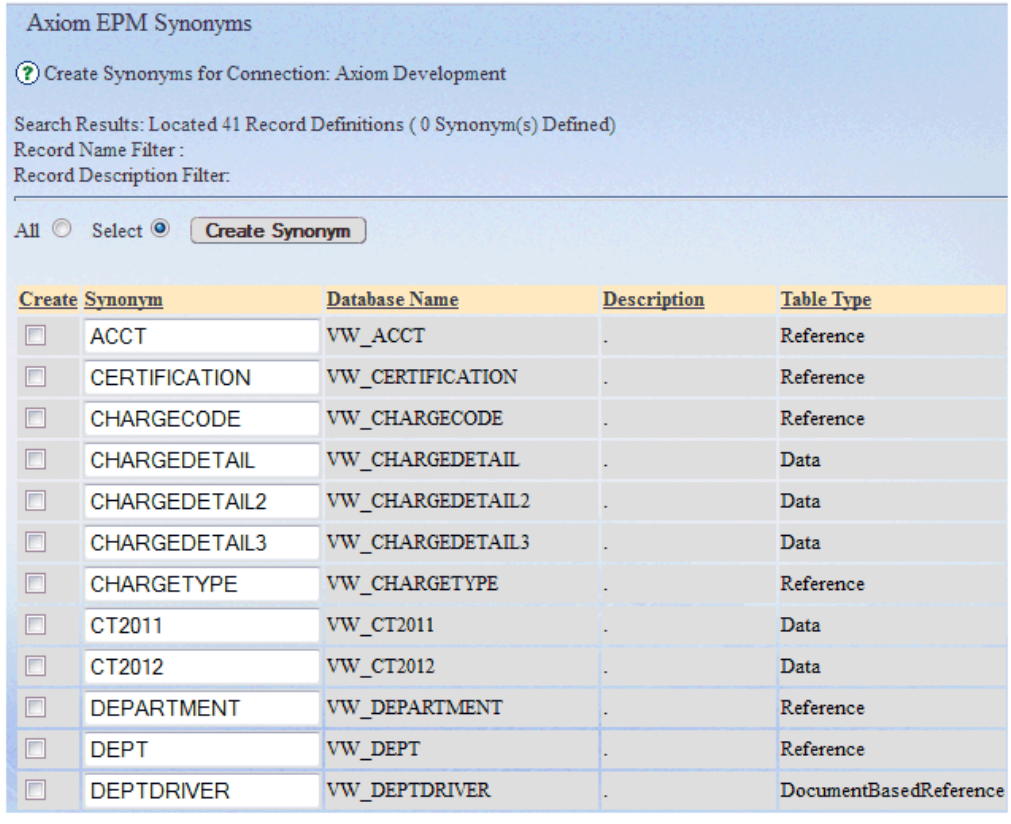
Axiom EPM Tables can be filtered by Name, Description, or Table Type. Provide filter values in the respective fields, or leave the fields blank to see a full list of tables.

Note:

- The Table Description is case-sensitive.
- If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

3. Click *Next* .

The Create Synonyms for Connection window opens.



Axiom EPM Synonyms

Create Synonyms for Connection: Axiom Development

Search Results: Located 41 Record Definitions (0 Synonym(s) Defined)
Record Name Filter :
Record Description Filter:

All Select Create Synonym

Create	Synonym	Database Name	Description	Table Type
	ACCT	VW_ACCT	.	Reference
	CERTIFICATION	VW_CERTIFICATION	.	Reference
	CHARGECODE	VW_CHARGECODE	.	Reference
	CHARGEDETAIL	VW_CHARGEDETAIL	.	Data
	CHARGEDETAIL2	VW_CHARGEDETAIL2	.	Data
	CHARGEDETAIL3	VW_CHARGEDETAIL3	.	Data
	CHARGETYPE	VW_CHARGETYPE	.	Reference
	CT2011	VW_CT2011	.	Data
	CT2012	VW_CT2012	.	Data
	DEPARTMENT	VW_DEPARTMENT	.	Reference
	DEPT	VW_DEPT	.	Reference
	DEPTDRIVER	VW_DEPTDRIVER	.	DocumentBasedReference

4. Select the check box in the *Create* column for all the Tables you want to create a synonym for, or select *All* to create a synonym for all available Tables.
5. Optionally, change the synonym name in the *Synonym* name column.
6. Click *Create Synonym* to process. Processing may take a few minutes for each selected Record Definition.

Removing Synonyms

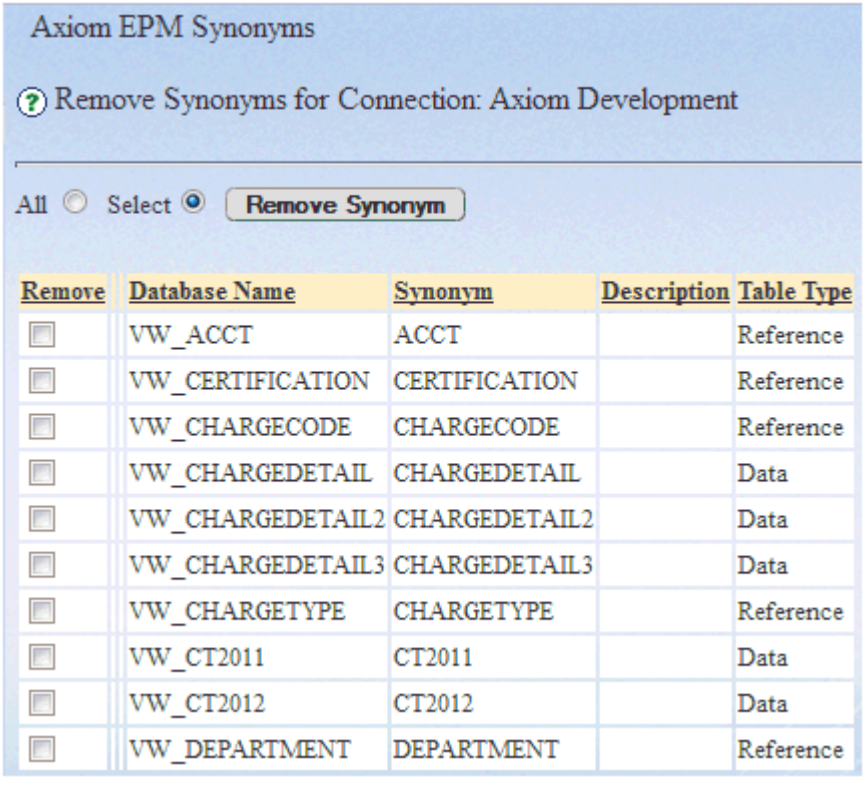
You can remove previously created Axiom EPM synonyms from the metadata catalog with the Remove Synonyms menu option.

Procedure: How to Remove Synonyms

1. In the Axiom EPM menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the Axiom EPM menu, see *Accessing the Adapter for Axiom EPM Administrator* on page 338.

2. Click *Remove* under the Synonyms group.

   The Remove Synonyms for Connection window opens.



Remove	Database Name	Synonym	Description	Table Type
	VW_ACCT	ACCT		Reference
	VW_CERTIFICATION	CERTIFICATION		Reference
	VW_CHARGECODE	CHARGECODE		Reference
	VW_CHARGEDETAIL	CHARGEDETAIL		Data
	VW_CHARGEDETAIL2	CHARGEDETAIL2		Data
	VW_CHARGEDETAIL3	CHARGEDETAIL3		Data
	VW_CHARGETYPE	CHARGETYPE		Reference
	VW_CT2011	CT2011		Data
	VW_CT2012	CT2012		Data
	VW_DEPARTMENT	DEPARTMENT		Reference

3. Select the check box in the *Remove* column for each synonym you want to remove, or select *All* to remove all available synonyms.
4. Click *Remove Synonym*.

Re-synchronizing Synonyms

You can re-synchronize your metadata. This option:

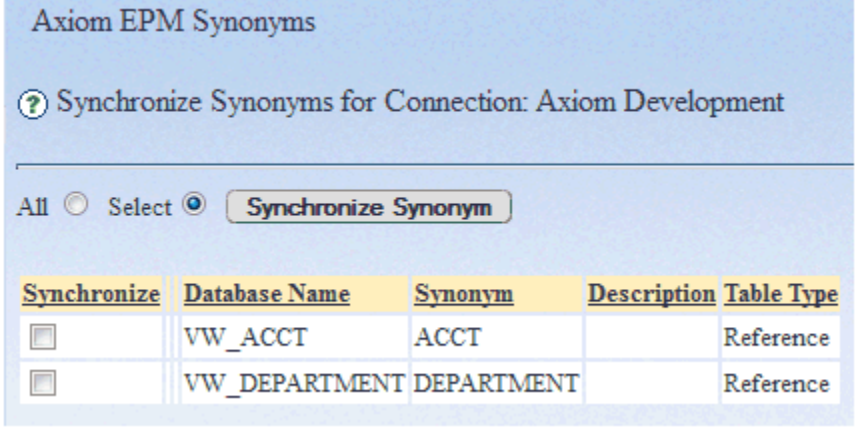
- Queries the internal repository to determine which records were installed for the selected Axiom EPM connection.
- Compares the modified date in those records with the modified date in the Axiom EPM data source. If none of the records have changed, a message appears.

If there are any records that have changed, they appear in the Select Record(s) to Synchronize page.

Procedure: How to Re-synchronize Synonyms

1. In the Axiom EPM menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the Axiom EPM menu, see *Accessing the Adapter for Axiom EPM Administrator* on page 338.
2. Click *Re-Synch* under the Synonyms group.

   The Synchronize Synonyms for Connection window opens.



3. Select the check box in the Synchronize column for each synonym you want to re-synchronize with Axiom EPM Record Definitions, or select *All* to re-synchronize all listed synonyms.
4. Click *Synchronize Synonym*.

Note: If any changes were made to the Axiom Synonym external of the adapter administration screen, the re-synchronize option will recreate the selected synonym(s) and the changes will be lost.

Refreshing Synonyms

After creating your synonyms, you may need to fresh them at some point.

Procedure: How to Refresh a Synonym

1. From the Axiom EPM menu, select the connection you want to manage from the *Select Connection* drop-down menu. For details on accessing the Axiom EPM menu, see *Accessing the Adapter for Axiom EPM Administrator* on page 338.

2. Click *Refresh* under the *Synonyms* group.

   The Refresh Synonyms for Connection window opens.

3. Select the synonyms you want to fresh under the *Refresh* column, or select the *All* option.
4. Click *Fresh Synonym*.

Note: If any changes were made to the Axiom Synonym external of the adapter administration screen, the re-fresh option will recreate the selected synonym(s) and the changes will be lost.

Viewing Sample Data

After creating a synonym, you may want to verify that it is using the correct Table. The *Synonyms*, *View* option on the Axiom EPM menu provides an easy method for listing the existing synonyms and viewing a sample data set from each.

Renaming a Synonym

You can selectively rename a synonym using the *Synonyms*, *Edit* option on the Axiom EPM menu.

After renaming a synonym, you must update references in existing reports from the old name to the new name using appropriate options in your reporting tool.

Managing Connections to Axiom EPM

Multiple connections to Axiom EPM data sources are possible from within one server configuration. However, depending on your reporting requirements, it may be optimal to create separate configurations for each data source.

Creating multiple connections using the same server configuration is a good option under the following circumstances:

- Two or more Axiom EPM data sources that must be accessed reside on the same system. If the data sources must be accessed for display in a single report output and if the server is located on the same computer system, this architecture can provide optimal performance.
- Two or more Axiom EPM data sources reside in different locations with requirements for moderate to significant consolidated reporting. This architecture provides optimal effectiveness in report development, system maintenance, and performance.

Create separate configurations instead of using multiple connections in these situations:

- Only a single Axiom EPM production data source must be supported. Each development and test instance should have a separate configuration.

- Multiple Axiom EPM data sources exist with only a limited requirement for combining data in reports.

The remaining instructions in this topic assume a single configuration directory structure for one or more data source connections.

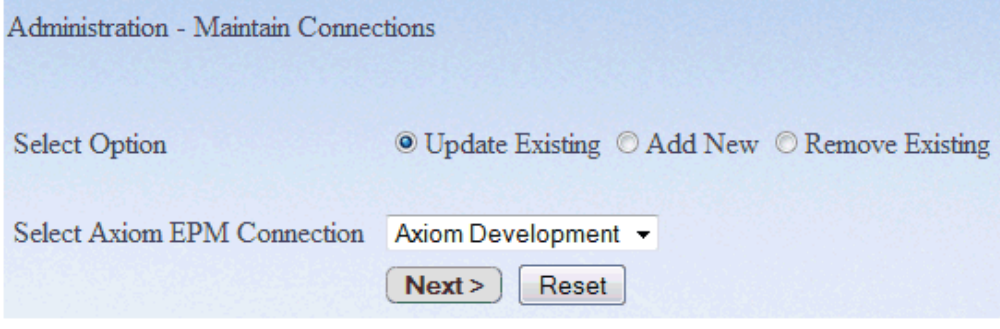
Updating an Axiom EPM Connection

If it becomes necessary to update the connection parameters to an Axiom EPM data source, you can use Administration options to perform these changes.

Procedure: How to Edit an Axiom EPM Connection

1. Click *Connections* under the *Administration* group.

   The Administration - Maintain Connections window opens.



2. Select *Update Existing*.
3. Select a connection from the *Select Axiom EPM Connection* drop-down list.
4. Click *Next*.

   The Administration - Update an Axiom EPM Connection window opens. For details, see *Administration - Create a New Axiom EPM Connection Window* on page 335.

5. Make the necessary modifications, test the connection, and if successful, click *Configure*. If not successful, edit the information until the test is successful, then click *Configure*. For more information, see step 4 in *How to Add an Axiom EPM Data Source Instance* on page 346.

   Note: The *Configure* button becomes available once the connection test to the selected RDBMS is successful.

6. On the next screen, click *Next*.

Adding an Axiom EPM Connection

After you add your first connection, you can specify additional Axiom EPM connections. Note that during this process, you are not prompted for the DBA information prior to managing metadata and connection information since all additional Axiom EPM data sources must use the same DBA in order for base metadata to function properly.

Procedure: How to Add an Axiom EPM Data Source Instance

1. Click *Connections* under the *Administration* group.

   The Administration - Maintain Connections window opens.

2. Select *Add New*.

   Note: If there are currently no connections, *Add* will be the only option.

3. Click *Next*.

   The Administration - Create a New Axiom EPM Connection window opens. For details, see *Administration - Create a New Axiom EPM Connection Window* on page 335.

4. Type the necessary information and click *Test Connection*. If

   - The connection is successful, the following message will appear in green to the right of the button:

     `Connection successful`

   - The connection is unsuccessful, the following message will appear in red to the right of the button with the relevant RDBMS return code:

     `Connection failed - return code: nnnn`

   When the test is successful, the *Configure* button appears.

5. Click *Configure*.
6. On the next screen, click *Next*.

When processing finishes, you can manage your synonyms for this connection.

Removing an Axiom EPM Connection

The integration with Axiom EPM depends on valid connection information. If a previously created connection is no longer required, you should remove that connection.

Procedure: How to Remove an Axiom EPM Connection

1. Click *Connections* under the *Administration* group.

The Administration - Maintain Connections window opens.

2. Select a connection from the *Select Connection* drop-down list.
3. Select *Remove Existing*.
4. Click *Next*.

   The Administration - Remove an Axiom EPM Connection window opens.

   You are prompted to confirm. After confirmation, the system removes the selected Axiom EPM Connection and all of its associated metadata.

Using Administrative Utilities

The Adapter for Axiom EPM provides administrative tools that enable you to report on connection information, update the DBA Password, and set adapter tracing options.

Connection Reports

The Administrator provides reports to make administration easier. You can run reports that provide information on connections and metadata. These reports are accessible from the Administration group menu.

Procedure: How to Run Connection Reports

1. Click *Reports* under the Administration group.

   The Administration - Reports window opens.

2. If necessary, select the Axiom EPM connection you want to run a report for from the *Axiom EPM Connection* drop-down list.

3. Select a report type from the Administration Report drop-down list. The options are:

   - **Connection Information.** Retrieves statistics for the selected connection.
   - **Synonyms.** Retrieves a list of available synonyms for the selected connection.

4. Select an output format for the report from the Output Format drop-down list. The options are HTML and PDF.
5. Click *Run*.

The selected report appears.

Updating the DBA Password

The DBA Password option is used for updating the password that is supplied during the initial configuration of the adapter. When the password is updated, all synonyms are updated as well. (Note that this process may take several minutes.)

Tracing Adapter Processing

The adapter tracing utility creates application level traces for support purposes. Typically, tracing is used under the direction of Information Builders Customer Support Services.

Log off

This option, when clicked, will provide a confirmation dialog box to log off from the adapter and free the server agent. Select *OK* or *Cancel*.

Note: Closing the window will have the same effect (without the confirmation box).

Chapter 13

Using the Adapter for C9 INC

The Adapter for C9 INC provides read access to Salesforce data stored in C9, and supports C9 temporal features by generating fields with temporal properties in the Master File. The adapter converts application requests into C9 INC calls and returns optimized answer sets to the requesting application.

For details about C9 INC-supported data sources, see the *Server Release Notes*.

In this chapter:

Preparing the C9 INC Environment

In order to use the Adapter for C9 INC, you must install the C9 INC driver for whichever data source you would like to access, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the C9 INC Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or Linux location for some C9 INC Driver files to /usr/certive/cjdbc_xx.jar, issue the commands:

   ```
   CLASSPATH=/certive/cjdbc_xx.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\certive\cjdbc_xx.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and checking that the special services section displays "JSCOM3 active".

Configuring the Adapter for C9 INC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to a C9 INC data source, the adapter requires connection and authentication information.

The connection credentials for a C9 INC data source are the credentials from the Salesforce account of the user.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one C9 INC data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for C9 INC

The *C9 INC* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the C9 INC data source:

```
jdbc:certive://host:port
```

Driver name

Name for the C9 INC driver:

```
com.certive.jdbc.CertiveDriver
```

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

The user is authenticated with Explicit authentication. This means that the user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.

User

Email address by which you are known to the data source.

Password

Password associated with the email address by which you are known to the data source.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLC9 SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

SQLC9

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name.

URL

Is the URL to the location of the C9 INC data source.

userid

Is the email address by which you are known to the target database.

password

Is the password associated with the email address by which you are known to the target database.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the C9 INC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLC9 SET CONNECTION_ATTRIBUTES CON1
  'jdbc:certive://host:port'
```

Configuring the Adapter

In order to connect to a C9 INC data source, the adapter requires connection and authentication information.

The connection credentials for a C9 INC data source are the credentials from the Salesforce account of the user.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one C9 INC data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLC9 SET DEFAULT_CONNECTION connection
```

where:

`SQLC9`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLC9 SET DEFAULT_CONNECTION SAMPLE
```

Managing C9 INC Metadata

For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLC9 to identify the Adapter for C9 INC.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLC9 [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLC9

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for C9 INC

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Dynamic columns

To specify that the Master File created for the synonym should not contain column information, select the *Dynamic columns* check box.

If this option is selected, column data is retrieved dynamically from the data source at the time of the request.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 365.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for C9 INC synonym comprises a Master File and an Access File. This is a synonym for the table account.

Master File account.mas

```
FILENAME=ACCOUNT, SUFFIX=SQLC9    , $
  SEGMENT=ACCOUNT, SEGTYPE=S0, $
    FIELDNAME=ID, ALIAS=Id, USAGE=A256V, ACTUAL=A256V,
      MISSING=ON, $
    FIELDNAME=ISDELETED, ALIAS=IsDeleted, USAGE=I11, ACTUAL=I4,
      MISSING=ON, $
    FIELDNAME=MASTERRECORDID, ALIAS=MasterRecordId, USAGE=A256V,
ACTUAL=A256V,
      MISSING=ON, $
    FIELDNAME=NAME, ALIAS=Name, USAGE=A256V, ACTUAL=A256V,
      MISSING=ON, $
    FIELDNAME=TYPE, ALIAS=Type, USAGE=A256V, ACTUAL=A256V,
      MISSING=ON, $
    FIELDNAME=PARENTID, ALIAS=ParentId, USAGE=A256V, ACTUAL=A256V,
      MISSING=ON, $
    FIELDNAME=BILLINGSTREET, ALIAS=BillingStreet, USAGE=A256V,
      ACTUAL=A256V, MISSING=ON, $
       .
       .
       .
    DEFINE DAILY_TREND/YYMD  WITH ID TEMPORAL_PROPERTY
TREND=DB_EXPR(INTERVAL '1' DAY);
      TITLE='Daily Trend', $
    DEFINE WEEKLY_TREND/YYMD  WITH ID TEMPORAL_PROPERTY
TREND=DB_EXPR(INTERVAL '1' WEEK);
      TITLE='Weekly Trend', $
    DEFINE MONTHLY_TREND/YYMD  WITH ID TEMPORAL_PROPERTY
TREND=DB_EXPR(INTERVAL '1' MONTH);
      TITLE='Monthly Trend', $
    DEFINE QUARTERLY_TREND/YYMD  WITH ID TEMPORAL_PROPERTY
TREND=DB_EXPR(INTERVAL '1' QUARTER);
      TITLE='Quarterly Trend', $
    DEFINE YEARLY_TREND/YYMD  WITH ID TEMPORAL_PROPERTY
TREND=DB_EXPR(INTERVAL '1' YEAR);
      TITLE='Yearly Trend', $
```

Access File account.acx

```
SEGNAME=ACCOUNT,
   TABLENAME=ads.Account,
   CONNECTION=CON01,
   KEY=ID, $
```

Reference: **Access File Keywords**

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the C9 INC table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* Absence of the CONNECTION attribute indicates access to the default database server.
KEY	Specifies the column that participates in the primary key. The value is always ID: KEY=ID
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: C9 Temporal Properties

The adapter supports C9 Temporal Analysis by generating fields with temporal properties in the Master File.

For example, by default, CREATE SYNONYM adds the following five fields to the Master File that enable the use of Temporal Analysis supported by C9:

```
DEFINE DAILY_TREND/YYMD WITH ID TEMPORAL_PROPERTY TREND = DB_EXPR(INTERVAL
'1' DAY); TITLE='Daily Trend', $
DEFINE WEEKLY_TREND/YYMD WITH ID TEMPORAL_PROPERTY TREND = DB_EXPR(INTERVAL
'1' WEEK); TITLE='Weekly Trend', $
DEFINE MONTHLY_TREND/YYMD WITH ID TEMPORAL_PROPERTY TREND =
DB_EXPR(INTERVAL '1' MONH); TITLE='Monthly Trend', $
DEFINE QUARTERLY_TREND/YYMD WITH ID TEMPORAL_PROPERTY TREND =
DB_EXPR(INTERVAL '1' QUARTER); TITLE='Quarterly Trend', $
DEFINE YEARLY_TREND/YYMD WITH ID TEMPORAL_PROPERTY TREND = DB_EXPR(INTERVAL
'1' YEAR); TITLE='Yearly Trend', $
```

By using one of these fields the user can perform temporal queries against the base tables. For example:

❑ Querying data as of a certain moment in the history of the table. The moment is specified using a WHERE equality test on any of the above fields.

The AS OF time itself can be added to the resulting report by referencing the same field in a BY clause. For example:

```
TABLE FILE OPPORTUNITY
WRITE AMOUNT WHERE DAILY_TREND EQ '2014-01-01'.
```

The resulting SQL query is the following:

```
AS OF DATE '2014-01-01' SELECT SUM(T1."Amount") FROM ads.Opportunity T1.
```

- Issuing a series of temporal queries within a specified range of dates with a specified interval, and plotting the results in a single report, thus obtaining the temporal trend of desired business performance indicators. For example:

```
TABLE FILE ibisamp/OPPORTUNITY WRITE AMOUNT BY QUARTERLY_TREND
WHERE QUARTERLY_TREND FROM '2013-02-01' TO '2015-03-31'
```

The resulting SQL query is the following:

```
TREND FROM DATE '2013-02-01' TO DATE '2015-03-31' BY INTERVAL '1' QUARTER
SELECT CAST(TrendDate() AS DATE), SUM(T1."Amount") FROM ads.Opportunity
T1 GROUP BY CAST(TrendDate() AS DATE) ORDER BY CAST(TrendDate() AS DATE)
```

Please note that instead of DATE constants in the WHERE clause above, you can use DATE expressions based on Dialogue Manager variables such &DATEYYMD, and simplified functions such as DTRUNC and DTADD.

You can specify a trend interval that is different from the five predefined in the synonym, and use it as in the following sample code:

```
DEFINE FILE ibisamp/OPPORTUNITY
BIWEEKLY_TREND/YYMD WITH ID TEMPORAL_PROPERTY TREND = DB_EXPR(INTERVAL 2
WEEK);
END
TABLE FILE ibisamp/OPPORTUNITY
COUNT ID BY BIWEEKLY_TREND ...
```

The following is an example of a Direct SQL Passthru temporal query:

```
SQL SQLC9   /*QUERY*/
 TREND FROM DATE '2013-01-01' TO CURRENT_DATE BY INTERVAL '1' QUARTER
   SELECT
  CAST(TrendDate() AS DATE),
   SUM(T1."Amount")
   FROM
  ads.Opportunity T1
   GROUP BY
  CAST(AsOfDate() AS DATE)
   ORDER BY
  CAST(AsOfDate() AS DATE);
END
```

Note: In order for a query against a C9 synonym with a temporal field (a field with TEMPORAL_PROPERTY TREND) to make business sense the following restrictions apply.

- Only one temporal field should be mentioned in a TABLE query.
- The field should be used in a sort phrase (BY or ACROSS).
- Temporal fields should not be objects of the WRITE or PRINT commands.
- Temporal fields should not be used in DEFINE or COMPUTE commands.
- If no temporal fields are referenced in a query, the results are returned as of the current date.
- It is recommended that the query contain a WHERE test on a temporal field. The following tests are allowed:
  - Single-point EQ,
  - Single-interval FROM-TO, GE, LE.

    Boundaries assumed by default, if not specified by a single-point EQ or FROM-TO test, are as follows:

    The lower boundary is the start of the UNIX epoch, that is, 01/01/1970.

    The upper boundary is the current date.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the following options.

Option	Description
Open	Opens the Master File for viewing and editing using a graphical interface. If an Access file is used it will be also available.
Edit as Text	Enables you to view and manually edit the Master File synonym. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the *Open* option, rather than manually editing the Master File.

Option	Description
Edit Access File as Text	Enables you to view and manually edit the Access File synonym. **Note:** This option is available only when an Access File is created as part of the synonym.
Sample Data	Retrieves up to 20 rows from the associated data source.
Data Profiling	Data Profiling provides the data characteristics for synonym columns. Alphanumeric columns provide the count of distinct values, total count, maximum, minimum, average length, and number of nulls. Numeric columns provide the count of distinct values, total count, maximum, minimum, average value, and number of nulls.
Refresh Synonym (if applicable)	Regenerates the synonym. Use this option if the underlying object has been altered.
Impact Analysis	Generates reports on procedures, synonyms, and columns that provide information on the flows/stored procedures available on a particular server, and the synonyms and columns they use. These reports enable you to evaluate changes before they are made by showing which components will be affected. See the *Server Administration* manual for details about Impact Analysis reports.
Copy	Copies the synonym to the clipboard.
Delete	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Cut	Deletes the synonym and places it on the clipboard.
Properties	Displays the properties of the synonym, including physical location, last modified date, description, and privileges.

Data Type Support Report

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Customizing the C9 INC Environment

The Adapter for C9 INC provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to C9 INC.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLC9 SET TIMEOUT {nn|0}
```

where:

`SQLC9`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native C9 INC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Chapter 14

Using the Adapter for Caché

The Adapter for Caché allows applications to access Caché data sources. The adapter converts application requests into native Caché statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Caché Environment (ODBC)

In order to use the adapter, the Caché client must be installed on the box where the server will be running.

Procedure: How to Set up the ODBC Environment on Windows

On Windows, the Caché environment is set up during the installation of the product.

Procedure: How to Set Up the ODBC Environment on UNIX

To set up the Adapter for Caché environment on UNIX, export the following environment variables:

- **Caché_HOME.** Specify the full name of the directory in which Caché is installed. For example:

  ```
  Cache_HOME=/rdbms/Cache
  ```

- **ODBCINI.** Specify the location of the ODBC initialization file. For example:

  ```
  ODBCINI=/odbms/Cache/Cachex.ini
  ```

Preparing the Caché Environment (JDBC)

In order to use the Adapter for JDBC, you must install the JDBC driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the JDBC Environment on Windows and UNIX

1. Identify the location of the JDBC Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;
   $CLASSPATH
   export CLASSPATH
   ```

   On UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

2. Start, or restart, the server.

   You must also restart the server if the driver has changed.

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace and checking that the special services section displays *JSCOM3 active*.

   ```
   edastart -show
   ```

   If the JSCOM3 service is not active, a *fail to start* message will typically also be inserted in the server EDAPRINT log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for Caché

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to the Caché database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Caché database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Caché Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Caché (ODBC)

The *Caché* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

Caché data source name (DSN). This is no default data source name. You must enter a value.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually (ODBC)

```
ENGINE SQLISM SET CONNECTION_ATTRIBUTES connection
       DSN_name/userid,password
```

where:

`SQLISM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`userid`

Is the primary authorization ID by which you are known to the adapter.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the Caché DSN named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLISM SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

Reference: Connection Attributes for Caché (JDBC)

The Cache adapter is under the SQL group folder. The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Is a logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL pointing to the location of the JDBC data source.

Driver name

Is the name of the JDBC driver.

For information, see the driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names that will be available for Java Services. This value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field.

When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Syntax: How to Declare Connection Attributes Manually (JDBC)

```
ENGINE SQLISM SET CONNECTION_ATTRIBUTES connection 'url'/userid,password
```

where:

`SQLISM`
Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`
Is the connection name.

`'url'`
Is the URL to the location of the JDBC data source.

`userid`
Is the primary authorization ID by which you are known to the target database.

`password`
Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes (JDBC)

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

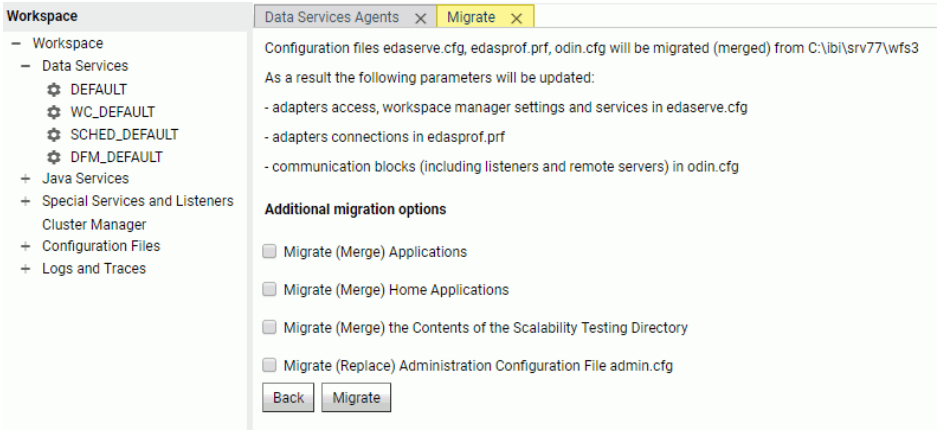
```
ENGINE SQLISM SET CONNECTION_ATTRIBUTES CON1
'jdbc:xxxxxxx://hostname:port/datasource'/MYUSER,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to a newer release. To access this option, click *Workspace* from the sidebar, then *Migrate* on the ribbon, or right-click *Workspace* on the resources tree, then click *Migrate* on the context menu. This is the recommended approach.

On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click *Continue*. Select additional migration options, as shown in the following image, and click *Migrate*



If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior to Version 7 Release 6.1, the connection was defined as:

  ```
  ENGINE SQLISM SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

  ```
  ENGINE SQLISM SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
  ```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLISM SET DEFAULT_CONNECTION connection
```

where:

`SQLISM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLISM SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLISM SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLISM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Caché Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Caché data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLISM to identify the Adapter for Caché.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLISM [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLISM

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using Caché, you must locally install the RDBMS Caché Driver.

The Server can access third-party database tables across the network Caché. You must provide a system data source name and possibly a user ID and/or password for the database tables you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each Caché table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.

- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Caché

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Caché synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLISM ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Caché table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local Caché database server. Absence of the CONNECTION attribute indicates access to the default database server.
DBSPACE	Optional keyword that indicates the storage area for the table. For example: *datasource.tablespace* DATABASE *datasource*
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.

Keyword	Description
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Caché Environment

The Adapter for Caché provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Caché.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLISM SET TIMEOUT {nn|0}
```

where:

`SQLISM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLISM SET PASSRECS {ON|OFF}
```

where:

`SQLISM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE SQLISM SET ISOLATION {RU|RC|RR|SE}
```

where:

`SQLISM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`RU`

Sets the transaction isolation level to Read Uncommitted.

`RC`

Sets the transaction isolation level to Read Committed.

`RR`

Sets the transaction isolation level to Repeatable Read.

`SE`

Sets the transaction isolation level to Serializable Read.

Caché Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 15

Using the Adapter for CICS Transactions

The Adapter for CICS Transactions processes input parameters, creates and sends requests to the CICS Transaction Server, and creates an answer set based on the received response.

This adapter supports the following transport layers:

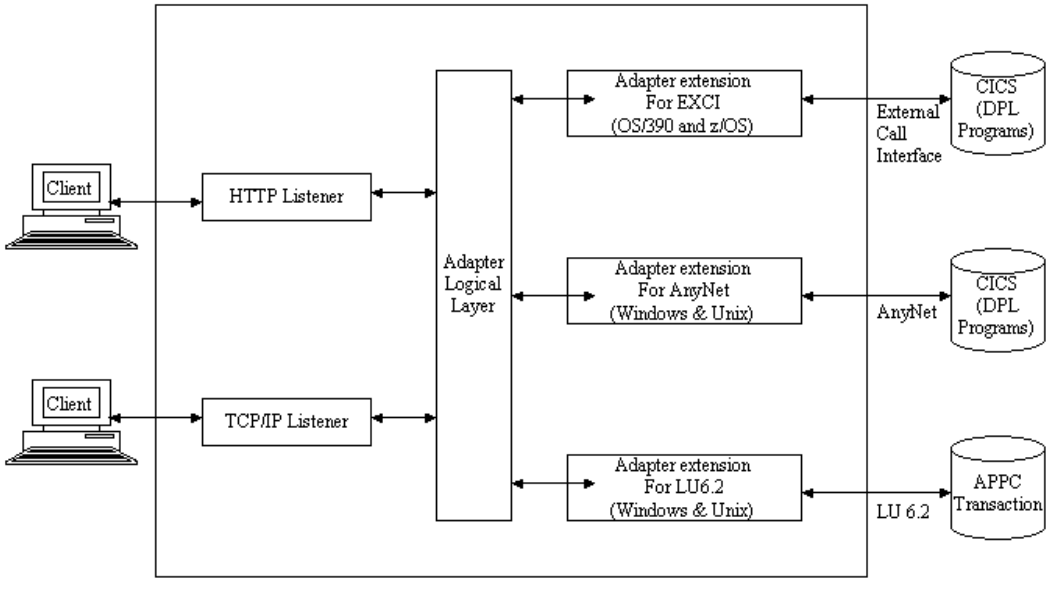
- EXCI for z/OS platforms.
- TCP62 using AnyNET for UNIX and Microsoft Windows platforms.
- TCP/IP for UNIX, Microsoft Windows, and z/OS platforms.
- LU6.2 for UNIX, Microsoft Windows, and z/OS platforms.

Note: Throughout this chapter, the CICS Transaction Server is referred to as CICS Transactions.

In this chapter:

Preparing the CICS Environment

The following diagram illustrates the basic flow of a CICS transaction. It illustrates how the client application communicates with the adapter.



The communication to the CICS Transactions region depends upon the platform the adapter is running on. If you are using UNIX or Microsoft Windows, you can use a communication transport of TCP62 (AnyNET) for DPL programs or LU6.2 for APPC programs. If you are using z/OS, a direct CICS Transactions External Call Interface is used for DPL programs (use of LU6.2 for APPC programs from z/OS is not supported in this release of the server).

To provide for transparent execution of CICS Transactions programs, metadata describing the program input and output areas is held on the server. For DPL programs, the COBOL FD describing DFHCOMMAREA is used as a source for metadata creation. For APPC-based programs, the COBOL FD describing the storage layout for CICS Transactions SEND and RECEIVE calls is used as a source for metadata creation; RECEIVE for input, SEND for output.

A Web Console is available as the primary configuration and maintenance tool. It is used to add or change adapter communication parameters and in creating/maintaining the metadata associated with the programs to be executed.

CICS Transactions Adapter Supported Platforms and Release Information

The following platforms are supported: Microsoft Windows, UNIX, and z/OS.

For Microsoft Windows and UNIX based servers that use AnyNET or LU6.2 to communicate with any CICS Transactions region, the following minimum software release levels on a target are required:

- z/OS Version 1.4 or higher.
- CICS Transactions Version 4 or higher.
- TCP/IP Direct Access, CICS TS 2.3 or higher.

Note: To use TCP62 or AnyNET, all VTAM options for AnyNET or TCP62 must be configured and AnyNET must be active.

For z/OS based servers, the following minimum software release levels are required:

- z/OS Version 1.4 or higher
- CICS Transactions Version 4 or higher

Note: Because the EXCI option is being used, the adapter can only communicate with CICS Transactions regions that are running on the same LPAR. Also, the STEPLIB of the IRUNJCL JCL member of the configuration data set needs to include the CICS Transactions SDFHEXCI library.

CICS and VTAM Configuration

These topics provide the setup and communications configuration information the adapter requires to communicate with CICS Transactions.

After you configure any VTAM definitions, the major nodes they describe need to be started in the VTAM system. If you are using AnyNET, it should be tested from the Adapter for CICS Transactions platform.

AnyNET VTAM Definitions

The AnyNET product requires three major VTAM nodes for its operation. The following are examples of those nodes. Replace relevant parameters with your site specific values.

Example: **Setting Up the Major Node for the AnyNET Listener**

```
**********  TCP62 MAJOR NODE DEFINITION FOR ANYNET
TCP62   VBUILD TYPE=TCP,DNSUFX=IBI.COM,PORT=397
TCP62G  GROUP  ISTATUS=ACTIVE
TCP62L  LINE   ISTATUS=ACTIVE
TCP62P  PU     ISTATUS=ACTIVE,NETID=MYNET
```

Note: Port 397 is used in this example. On UNIX, using a port number above 1000 is recommended due to the root privileges associated with using a port number under 1000. On z/OS, 397 is the default port. It is well established and frequently used.

Keep in mind, however, that the port you use must be the same on UNIX and z/OS. For example, if 1000 is specified on one platform, it must also be specified on the other.

Example: Setting Up the Major Node for Cross Domain Resources

If CDRDYN=YES is not specified in the ATCSTR00 VTAM startup parameter file, then the name of the machine that the Transaction adapter is running on needs to be coded in the follow node:

```
        VBUILD TYPE=CDRSC
CDRSC62 NETWORK NETID=MYNET
CDRSC62 GROUP
EDABGR2 CDRSC ALSLIST=TCP62P
```

Note: EDABGR2 is the machine name on which the server is running. The machine name must be limited to eight bytes.

Example: Setting Up the Standard VTAM LU Definitions

```
        VBUILD TYPE=SWNET
MYNAMEPU  PU   ADDR=01,IDBLK=05D,IDNUM=10101,                X
               MAXPATH=3,                                    X
               PUTYPE=2,                                     X
               MODETAB=MTOS2EE,                              X
               DLOGMOD=PARALLEL,                             X
               ISTATUS=ACTIVE
MYLUNAME  LU   LOCADDR=2
```

LU6.2 VTAM Definitions

LU6.2 requires two major VTAM nodes for its operation. The main node for AnyNET identifies the port. The following are examples of these two nodes. Replace relevant parameters with your site specific values.

Example: Setting Up the Major Node for the AnyNET Listener

```
**********  TCP62 MAJOR NODE DEFINITION FOR ANYNET
TCP62   VBUILD TYPE=TCP,DNSUFX=IBI.COM,PORT=397
TCP62G  GROUP  ISTATUS=ACTIVE
TCP62L  LINE   ISTATUS=ACTIVE
TCP62P  PU     ISTATUS=ACTIVE,NETID=MYNET
```

Note: Port 397 is used in this example. On UNIX, using a port number above 1000 is recommended due to the root privileges associated with using a port number under 1000. On z/OS, 397 is the default port. It is well established and frequently used.

Keep in mind, however, that the port you use must be the same on UNIX and z/OS. For example, if 1000 is specified on one platform, it must also be specified on the other.

Example: Setting Up the Standard VTAM LU Definitions

```
        VBUILD TYPE=SWNET
MYNAMEPU  PU   ADDR=01,IDBLK=05D,IDNUM=10101,                X
               MAXPATH=3,                                    X
               PUTYPE=2,                                     X
               MODETAB=MTOS2EE,                              X
               DLOGMOD=PARALLEL,                             X
               ISTATUS=ACTIVE
MYLUNAME  LU   LOCADDR=2
```

CICS Connection and Sessions for Microsoft Windows and UNIX

For the adapter to be able to connect to CICS (for both TCP62 and LU6.2 where supported), connection and session definitions are required. The definitions are dependent on the platform on which the adapter is running.

If you deploy the adapter on Microsoft Windows or UNIX, the connection and sessions definitions described in *Using the CEDA View Connection Command* on page 391 and *Using the CEDA View Sessions Command* on page 392 are required.

Note: For DPL program execution, all users must be able to access and run transaction CPMI (the mirror transaction).

Example: Using the CEDA View Connection Command

The following is an example of a CEDA view connection command that illustrates what is required:

```
CEDA  View Connection(  MYLU )
   Connection    :  MYLU
   Netname       :  MYLUNAME
   ACcessmethod  :  Vtam          Vtam  | IRc  | INdirect | Xm
   PRotocol      :  Appc          Appc  | Lu61 | Exci
   SInglesess    :  No            No    | Yes
   DAtastream    :  User          User  | 3270 | SCs | STrfield | Lms
   INService     :  Yes           Yes   | No
   ATtachsec     :  Verify        Local | Identify | Verify | Persistent
```

Example: Using the CEDA View Sessions Command

The following is an example of a CEDA view sessions command that illustrates what is required:

```
CEDA View Sessions(  MYLUNAME )
   Sessions      :  MYLUNAME
   Connection    :  MYLU
   MOdename      :  PARALLEL
   Protocol      :  Appc               Appc | Lu61 |
   MAximum       :  008 , 000          0-999
```

The MAximum parameter determines the number of concurrent sessions available to process requests. Its first value should be in the range of 4 - 255, and the second one should always be set to zero.

Note: Log mode must support parallel sessions.

Configuring the Adapter for CICS Transactions

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. In addition, for if you are using the TCP 62 or LU 6.2 communication protocols, you must configure a Listener node.

Configuring Communications Parameters for TCP 62 and LU 6.2

This procedure is required if you are using TCP 62 or LU 6.2 as your communications protocol.

Note that you can complete this step either *before* or *after* you declare connection attributes.

Procedure: How to Configure a Listener

Note: This step is required if you are using TCP 62 or LU 6.2 on Windows or UNIX.

1. From the Web Console menu bar, select *Workspace* then *Configuration*.
2. On the expanded menu bar, select *New*, then *Listener*, and *CICST*. The Listener Configuration pane opens.
3. Type the appropriate values for the communications parameter.

   For descriptions of these parameters, see *Communications Parameters for CICS Transactions for TCP 62 or LU 6.2* on page 393.

   Tip: You can also obtain details about each entry by clicking the *?* icon to the left of any parameter field on the communications configuration pane to access the Web Console Help.
4. Click *Save and Restart*.

Reference: **Communications Parameters for CICS Transactions for TCP 62 or LU 6.2**

This chart describes the parameters required to configure communications. To complete the configuration, click *Save and Restart Listener*.

Keyword	Description
NODE	8-character string Defines a logical name of a node block, which must be unique in the file. The logical name can be a maximum of eight characters, must begin with a letter, and can include any characters, except semicolon and equal sign. A node block contains keyword-value pairs, which are enclosed in a BEGIN-END statement.
PARTNER_LU_NAME	string Defines the APPLID of the target CICS region.
LOCAL_LU_NAME	string Defines the LU name to be used. This value is taken from an LU entry in the major node. The value is also specified in the CICS connection/sessions definition.
MODE_NAME	string Defines the DLOGMODE parameter value.
CODEPAGE	string Defines the code page that the CICS region is using.
COMMUNICATION	Select one of the following options: ❑ TCP/IP for DPL program execution. This option displays additional parameters required for TCP 62. ❑ LU6.2 for APPC program execution. **Important:** Your selection in this field determines what additional configuration information that is required.
Additional required information for TCP/IP (This option displays additional parameters required for TCP 62.)	

Keyword	Description
VTAM_NETWORK_ID	string Defines the VTAM network ID. This value is taken from the AnyNET listener major node under parameter NETID.
CONNECT_LIMIT	number of seconds Defines the maximum time, in seconds, that the client waits for a connection response from the AnyNET environment. Provide a value greater than 0. (Do not enter a value of -1.)
PROXY	string Defines the name of the supplied proxy (user-written) program, which converts the Full Function Server calling syntax to the syntax that is valid for the CICS program to be executed. This parameter is *optional*, and should be left blank for normal operations. Use it only at the direction of your local support personnel.
MIRROR	string Defines the IBM-supplied mirror transaction CPMI that the Adapter for CICS Transactions calls by default. If this transaction cannot be used, you can create another transaction name to point to the IBM-supplied mirror program DFHMIRS. Enter the alternate name in the MIRROR field.
NATPROG	string Identifies the NATURAL program to be executed. **Note:** This option does not apply to the Adapter for CICS Transactions.
HOST	string Defines the host that a client is connecting to or an IP address that a listener is listening on.

Keyword	Description
SERVER_ID	string Defines the machine name where the server is configured with the CICS AGENT NODE (Listener).
ANYNET_PORT	positive integer number Defines the port number that the AnyNET product is using. The default AnyNET port is 397, but any number can be used. For UNIX installations, a port number above 1000 is recommended.
Additional required information for LU6.2	
PROXY	string Defines the name of the supplied proxy (user-written) program, which converts the Full Function Server calling syntax to the syntax that is valid for the CICS program to be executed. This parameter is *optional*, and should be left blank for normal operations. Use it only at the direction of your local support personnel.
SIDE_INFO	string SIDE_INFO is a table with definitions for all partners to the currently active CICS system. CICS implements the side information table by means of the PARTNER resource. Partner resources are defined by the CEDA DEFINE command. To become known to an active CICS system, a defined partner resource must be installed using the CEDA INSTALL command.
HOST	string Defines the DNS name of the host LPAR of the target CICS region. You can also code the four part IP address.

Example: Sample Node Definitions Using LU6.2

```
NODE = CICSTLU6
BEGIN
  PROTOCOL = CICS
  CLASS = CICSCLIENT
  TARGET = CICS
  PARTNER_LU_NAME = EDBGM010
  LOCAL_LU_NAME = T29DPB41
  MODE_NAME = PARALLEL
  HOST = IBIMVS
  CODEPAGE = 037
  COMMUNICATION = LU62
END
```

Example: Sample Node Definitions Using TCP/IP (TCP 6.2)

```
NODE = LST_CICS
BEGIN
  PROTOCOL = CICS
  PORT = 8139
  CLASS = AGENT
END
NODE = SPGCSRV
BEGIN
  PROTOCOL = CICS
  PORT = 8139
  CLASS = AGENT
  TARGET = INTERNAL
  HOST = IBIMVS
  CODEPAGE = 037
  ANYNET_PORT = 397
END
NODE = CICSCLNT

BEGIN
  PROTOCOL = CICS
  CLASS = CLIENT
  TARGET = CICS
  INTEGER = HOBF
  FLOAT = IBM
  PARTNER_LU_NAME = EDBGM010
  LOCAL_LU_NAME = T29DPB04
  MODE_NAME = PARALLEL
  HOST = SPGCSRV
  CODEPAGE = 037
  COMMUNICATION = TCP
  VTAM_NETWORK_ID = USIBINET
  CONNECT_LIMIT = 60
  MIRROR = CPMI
  ATTACH_SECURITY = VERIFY
  SECURITY = C2B591CC2A92650028E8A570F3BE36F3
END
```

Configuring the Adapter

In order to connect to CICS Transactions, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to CICS Transactions takes place when the first query that references that connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the last SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the last SET CONNECTION_ATTRIBUTES command.

For detailed instructions about declaring connection attributes for your operating system, see *Configuring the Adapter on Microsoft Windows and UNIX* on page 397 or *Configuring the Adapter on z/OS* on page 401.

Configuring the Adapter on Microsoft Windows and UNIX

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. In addition, if you are using the TCP 62 or LU 6.2 communications protocol, you must configure a Listener node as described in *How to Configure a Listener* on page 392. For TCP/IP, you only need to declare connection attributes.

Procedure: How to Configure the Adapter on Windows and UNIX

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Select a communications protocol: *TCP 62*, *LU 6.2*, or *TCP/IP* and click *Next*.

   For TCP/IP, the configuration parameters for the adapter are displayed.

   For TCP 62 and LU 6.2, you are prompted to either *Configure CICST listener* or to click *Next* to display the configuration parameters for the adapter. You can complete these tasks *in either order*, but both must be done to use the adapter.

   - If you click *Next*, the configuration parameters are displayed. However, in this scenario, you must then configure a Listener node, as described in *How to Configure a Listener* on page 392.
   - If you click Configure CICST listener, the Web Console opens at the Special Services pane. For instructions, see *How to Configure a Listener* on page 392.

     Once you have configured the Listener, return to the Add CICS Transaction to Configuration pane, where the Listener node is now available for selection.
6. Enter values for the parameters required by the adapter as described in the connection attributes reference.
7. Click *Configure*. The configured adapter is added to the Adapters list in the navigation pane.

Reference: Connection Attributes for CICS Transactions on Windows and UNIX

The CICS Transaction adapter is under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Communication Protocol

Select one of the following: TCP/IP, TCP 62, LU 6.2. Subsequent options vary slightly depending on this selection.

Note: For TCP 62 and LU 6.2, you must configure a Listener either before or after you declare connection attributes.

This selection is reflected in the Communication field on the second configuration pane.

Connection name

Is a logical name used to identify a specific set of connection attributes.

Node name

For TCP 62 or LU 6.2, a drop-down list of nodes is displayed if the Listener has already been configured. Select a node from the list.

Host name

For TCP/IP, enter the name of the host machine.

Application ID

For TCP 62 or LU 6.2, enter the application ID of the CICS Transactions region.

Mirror

Enter the name of the mirror transaction.

Port

For TCP/IP, enter the port number.

Security

There are two methods by which a user can be authenticated when connecting to the Adapter for CICS Transactions:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the adapter, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to CICS Transactions, at connection time, for authentication.

User

Is the user name by which you are known to the adapter. This field is only displayed when the security mode of the server is non-trusted.

Password

Is the password associated with the user name. This field is only displayed when the security mode of the server is non-trusted.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

Syntax: How to Declare Connection Attributes Manually on Windows and UNIX

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES [connection]
  [node_name]/userid,password: "parameters_list"
```

Password passthru. The user ID and password are explicitly specified for each connection and passed to CICS Transactions, at connection time, for authentication.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES [connection]
  [node_name] /: "parameters_list"
```

where:

`CICSTRAN`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. See *Access File Attributes* on page 411.

`node_name`

Is a name of the NODE definition in the ODIN configuration file.

`userid`

Is the primary authorization ID by which you are known to the Adapter for CICS Transactions.

`password`

Is the password associated with the primary authorization ID.

parameters_list

The following parameters are supported (some required; some optional):

application_ID is the application ID of the CICS region. This parameter is optional.

mirror_name is the name of the mirror transaction. This value overwrites the MIRROR value provided when the Listener is configured. This parameter is optional.

communication is the communication protocol being used. The options are: TCP/IP, TCP 6.2 (AnyNET), LU 6.2.

Configuring the Adapter on z/OS

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Adapter on z/OS

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Select a communications protocol: *TCP/IP* or *EXCI* and click *Next*.

   The configuration parameters for the adapter are displayed.

6. Enter values for the parameters required by the adapter as described in the connection attributes reference.

7. Click *Configure*. The configured adapter is added to the Adapters list in the navigation pane.

Reference: Connection Attributes for CICS Transactions on z/OS

The CICS Transaction adapter in under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Communication Protocol

Select one of the following: *TCP/IP* or *EXCI*. Subsequent options vary slightly depending on this selection.

This selection is reflected in the Communication field on the second configuration pane.

Connection name

Is a logical name used to identify a specific set of connection attributes.

Host name

For TCP/IP, enter the name of the host machine.

Application ID

For EXCI, enter the application ID of the CICS region.

Port

For TCP/IP, enter the port number.

Mirror

Is the name of the mirror transaction. The default value is EXCI.

This value overwrites the value provided in the Listener mode.

Security

For EXCI, Trusted security is supported under the following conditions: the server must be running with security OPSYS and IRC (Inter Region Communications) must be active in the CICS region. The adapter then connects to the CICS region using the credentials of the operating system user impersonated by the server data access agent.

For TCP/IP, there are two methods by which a user can be authenticated when connecting to the Adapter for CICS Transactions:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the adapter, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to CICS Transactions, at connection time, for authentication.

User

Is the user name by which you are known to the adapter. This field is only displayed when the security mode of the server is non-trusted.

Password

Is the password associated with the user name. This field is only displayed when the security mode of the server is non-trusted.

Select profile

The level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)

Syntax: How to Declare Connection Attributes Manually on z/OS

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES [connection]
  [node_name]/userid,password: "parameters_list"
```

Password passthru. The user ID and password are explicitly specified for each connection and passed to CICS Transactions, at connection time, for authentication.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES [connection]
  [node_name] /: "parameters_list"
```

Trusted. The adapter connects to an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE CICSTRAN SET CONNECTION_ATTRIBUTES [connection]
  [node_name] /: "parameters_list"
```

where:

`CICSTRAN`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. See *Access File Attributes* on page 411.

node_name

Is a name of the NODE definition in the ODIN configuration file.

userid

Is the primary authorization ID by which you are known to the Adapter for CICS Transactions.

password

Is the password associated with the primary authorization ID.

parameters_list

The following parameters are supported (some required; some optional):

application_ID is the application ID of the CICS region. This parameter is optional.

mirror_name is the name of the mirror transaction. This value overwrites the MIRROR value provided when the Listener is configured. This parameter is optional.

communication is the communication protocol being used. The options are: TCP/IP and EXCI.

Managing CICS Transaction Metadata

When the server invokes a transaction or procedure, it needs to know how to build the request, what parameters to pass, and how to format an answer set from the response. For each transaction the server will execute, you must create a synonym that describes the layout of the request/response area.

Creating Synonyms

Synonyms define unique names (or aliases) for each transaction or procedure that is accessible from the server. Synonyms are useful because they hide the underlying transaction or procedure from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows the input parameters and the response layout to be moved while allowing client applications to continue functioning without modification. For example, moving a transaction or procedure from a test region to production. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Web Console or the Data Management Console.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the synonym creation parameters reference.

   - For Windows and UNIX, see *Synonym Creation Parameters for CICS Transactions on Windows and UNIX* on page 406.
   - For z/OS, see *Synonym Creation Parameters for CICS Transactions on z/OS* on page 408.
4. After entering the parameter values, click *Create Synonym*.

   The Status pane indicates that the synonym was created successfully.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the *Validate* check box, the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: **Synonym Creation Parameters for CICS Transactions on Windows and UNIX**

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Collection of COBOL definitions

Select *Collection of COBOL* definitions and type the location of a collection of Cobol Copybooks. If you select this option you can choose all COBOL definitions in the collection (%) or filter by name and extension.

File name/File extension

If you wish to limit retrieval, you can enter a file name and/or file extension:

- In the File name box, enter a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.
- In the File extension box, enter an extension with or without the wildcard symbol %.

Synonym name

Type the name of the synonym in the Synonym name box.

Program name

Type the name of the CICS Transactions program or APPC transaction in the Program name box.

COMMAREA

Enter a COMMAREA value: type CALC in the input box to indicate maximal segment size or enter a value that represents the physical length of the data. The default value is 32500.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select tables for synonyms

The Cobol Copybook(s) appear on the screen. You can choose the same Copybook or different Copybooks for input and output parameters.

Note: There can be no more than one set of input/output Copybooks per synonym.

Reference: Synonym Creation Parameters for CICS Transactions on z/OS

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

File System Selection

Choose one of the following options from the drop-down list:

- *Fully qualified PDS name* to indicate a partitioned data set on MVS.

  In the input boxes provided, type a PDS name preceded by // and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

  or

- *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

  In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and a File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).

Target Program type

Choose an option from the drop-down list:

- Dynamic Program Link
- TP Gateway (for migration)
- Stored Procedure Gateway (for migration)

Note: The last two options support older style program types (TPG/SPG/ASS) for which migration may be necessary. For details about migration, see the *ServerRelease Notes*.

COBOL FD available for the answer set

Check this box if you want to use COBOL file descriptions for your Master Files.

Synonym name

Type the name of the synonym in the Synonym name box.

Program name

Type the name of the CICS Transactions program or APPC transaction in the Program name box.

COMMAREA

Enter a COMMAREA value: type CALC in the input box to indicate maximal segment size or enter a value that represents the physical length of the data. The default value is 32500.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Input Segment Definition

Is the name of the COBOL FD file that contains information about input parameters.

There can be no more than one set of input COBOL Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

Output Segment Definition

Is the name of the COBOL FD file that contains information about output parameters.

There can be no more than one set of output Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Master File Guidelines

Master Files contain definitions for input and output parameters used in the transaction. These parameters are described as separate segments.

- A dummy segment must be defined if input parameters are not used.
- Output parameters (if they exist) are defined in a descendant segment/segments.
- Fields from output segments that contain repeating values can be redefined as an Occurs segment. A value for the keyword POSITION defines the redefined field name.
- A multi-segment definition with the RECTYPE field is used when a transaction may return different answer sets.

Example: Cobol Copybook for the ETPBRW8 Master File

```
02  FILEREC.
  03  STAT           PIC X.
  03  NUMB           PIC X(6).
  03  NAME           PIC X(20).
  03  ADDRX          PIC X(20).
  03  PHONE          PIC X(8).
  03  DATEX          PIC X(8).
  03  AMOUNT         PIC X(8).
  03  COMMENT        PIC X(9).
```

Example: Master File for the Transaction/Program

```
FILENAME=ETPBRW8, SUFFIX=CICSTRAN, CODEPAGE=37, $
  SEGMENT=SEG1, SEGTYPE=S0, $
   GROUP=FILEREC, USAGE=A80, ACTUAL=A80, $
    FIELDNAME=STAT, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=NUMB, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=NAME, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=ADDRX, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=PHONE, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=DATEX, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=AMOUNT, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=COMMENT, USAGE=A9, ACTUAL=A9, $
  SEGMENT=SEG11, SEGTYPE=S0, PARENT=SEG1, $
   GROUP=FILEREC, USAGE=A80, ACTUAL=A80, $
    FIELDNAME=STAT, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=NUMB, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=NAME, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=ADDRX, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=PHONE, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=DATEX, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=AMOUNT, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=COMMENT, USAGE=A9, ACTUAL=A9, $
```

Reference: Access File Attributes

Keyword	Description
SEGNAME	Is the name of the input segment in the Master File.
CONNECTION	Indicates the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. Defaults to the default connection.
TRANSACTION	Is the name of the program to be executed.

Keyword	Description
COMMAREA	Is the maximal size of COMMAREA for the Adapter for CICS Transactions. The options are: CALC means that the maximal segment length will be used. value defaults to 32500. You can enter a value that represents the physical length of the data.

Example: **Access File for Transaction/Program**

```
SEGNAME=SEG1,CONNECTION=CICST1,TRANSACTION=etpbrw8,COMMAREA=32500,
  FLOAT=IBM,INTEGER=BigEndian,$
```

Note: Do not change the FLOAT or INTEGER parameter values.

Example: **Sample CICS Transaction**

```
001700 IDENTIFICATION DIVISION.
001800 PROGRAM-ID. ETPBRW8.
001900 ENVIRONMENT DIVISION.
002000 DATA DIVISION.
002100 WORKING-STORAGE SECTION.
002200 01  .
002300     02  WS-RESPONSE            PIC 9(8) BINARY.
002500     02  WS-RECORD-SUB          PIC 9 VALUE 1.
002510      02  WS-ACC-NUM                PIC X(6).
002600 01  COMMAREA.
002601     05  MESSAGES-OUT           PIC X(1000).
002602*-------------------------------------------------------------*

002700 LINKAGE SECTION.
002800 01 DFHCOMMAREA.
003000     05  WS-OUTPUT-RECORD OCCURS 4 TIMES.
003100         10  STAT                 PIC X.
003110         10  ACC-NUM              PIC X(6).
003300         10  NAME                 PIC X(20).
003400         10  ADDRX                PIC X(20).
003500         10  PHONE                PIC X(8).
003600         10  DATEX                PIC X(8).
003700         10  AMOUNT               PIC X(8).
003800         10  COMMENTS             PIC X(9).
003900*-------------------------------------------------------------*
```

```
004000 PROCEDURE DIVISION.
004100 PROCESS-INPUT.
004110     MOVE ACC-NUM(1) TO WS-ACC-NUM
004200     EXEC CICS STARTBR FILE('FILEA')
004300               RIDFLD(WS-ACC-NUM)
004400               GTEQ
004500     END-EXEC
004600     .
004700
004800        MOVE 1 TO WS-RECORD-SUB
004900
005000     PERFORM 4 TIMES
005100
005200     EXEC CICS READNEXT FILE('FILEA')
005300                        RIDFLD(WS-ACC-NUM)
005400                        INTO(WS-OUTPUT-RECORD(WS-RECORD-SUB))
005500                        RESP(WS-RESPONSE)
005600     END-EXEC
005700
005800        ADD 1 TO WS-RECORD-SUB
005900
006000     END-PERFORM.
006100*-----------------------------------------------------------------*

006200     EXEC CICS RETURN
006300     END-EXEC.
006400     GOBACK
006500     .
```

Invoking a CICS Transaction

To invoke a CICS transaction, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, an SQL SELECT statement, or an EX command. (DML is Information Builders internal retrieval language.) For information about the Data Manipulation Language, see the *Stored Procedure Reference* manual.

Syntax: How to Invoke a CICS Transaction Using TABLE

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the CICS transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

IF/WHERE

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke a CICS Transaction Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the CICS transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

`WHERE`

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

`in-parameter`

Is the name of an input parameter to which you want to pass a value.

`value`

Is the value you are passing to an input parameter.

Syntax: How to Invoke a CICS Transaction Using EX

```
EX [app_name_space/]synonym  1=parm1_val,..., N=parmN_val

EX [app_name_space/]synonym parm1_name=parm1_val,... ,
                            parmN_name=parmN_val

EX [app_name_space/]synonym [, parm1_val [,...[, parmN_val]]]
```

where:

`app_name_space`

Is the apps directory name under which the synonyms are stored. This value is optional.

`synonym`

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

`parm_name`

Is the name of the parameter taken from the input metadata description.

`1=parm1_val`

`N=parmN_val`

`parm1_name`

`parm1_val`

`parmN_val`

Are the parameter values that match the input metadata description.

Note:

- Consecutive commas denote missing parameters in the positional form of the request.

- Values containing special characters (<equal sign> <space> <comma>) must be encapsulated in double or single quotation marks.

Example: **Invoking the ETPBRW8 Transaction**

The following TABLE command invokes the ETPBRW8 transaction, passing one parameter value to it.

```
TABLE FILE ETPBRW8
  PRINT SEG11.NUMB SEG11.STAT SEG11.NAME SEG11.ADDRX SEG11.PHONE
  IF SEG1.NUMB EQ '000700'
END
```

The output is:

```
NUMBER OF RECORDS IN TABLE=        4  LINES=      4


PAGE     1


NUMB    STAT  NAME                 ADDRX                PHONE
----    ----  ----                 -----                -----
000762        SUSAN MALAIKA        SAN JOSE,CALIFORNIA  22312121
000983        J. S. TILLING        WASHINGTON, DC       34512120
001222        D.J.VOWLES           BOBLINGEN, GERMANY   70315551
001781        TINA J YOUNG         SINDELFINGEN,GERMANY 70319990
```

The following SELECT command invokes the ETPBRW8 transaction:

```
SQL
SELECT  SEG11.NUMB, SEG11.STAT, SEG11.NAME, SEG11.ADDRX, SEG11.PHONE
FROM ETPBRW8
WHERE SEG1.NUMB = '000700';
END
```

The output is:

```
PAGE     1

NUMBER OF RECORDS IN TABLE=        4  LINES=      4

NUMB    STAT  NAME                 ADDRX                PHONE
----    ----  ----                 -----                -----
000762        SUSAN MALAIKA        SAN JOSE,CALIFORNIA  22312121
000983        J. S. TILLING        WASHINGTON, DC       34512120
001222        D.J.VOWLES           BOBLINGEN, GERMANY   70315551
001781        TINA J YOUNG         SINDELFINGEN,GERMANY 70319990
```

The following EX commands invoke the ETPBRW8 transaction:

```
EX ETPBRW8 '000700'

EX ETPBRW8 NUMB='000700'

EX ETPBRW8 1='000700'
```

The ETPBRW8 transaction browses the file starting from '000700' and, in these examples, returns values greater than '000700'.

Note: In order to use the EX command, you must set the SET EXORDER setting in either the FOCEXEC or in edasprof.prf.

```
SET EXORDER=PGM/FEX

EX baseapp/etpbrw8 NUMB='000700'
```

The output is:

```
FILE=SQLOUT,SUFFIX=FIX,$
  SEGNAME=SQLOUT,$
    FIELD=STAT,    E11       ,A1       ,A1       ,          ,$
    FIELD=NUMB,    E12       ,A6       ,A6       ,          ,$
    FIELD=NAME,    E13       ,A20      ,A20      ,          ,$
    FIELD=ADDRX,   E14       ,A20      ,A20      ,          ,$
    FIELD=PHONE,   E15       ,A8       ,A8       ,          ,$
    FIELD=DATEX,   E16       ,A8       ,A8       ,          ,$
    FIELD=AMOUNT,  E17       ,A8       ,A8       ,          ,$
    FIELD=COMMENT, E18       ,A9       ,A9       ,          ,$

000762SUSAN MALAIKA       SAN JOSE,CALIFORNIA 2231212101 06
74$0000.00*********

000983J. S. TILLING       WASHINGTON, DC      3451212021 04
75$9999.99*********

001222D.J.VOWLES          BOBLINGEN, GERMANY  7031555110 04
73$3349.99*********

001781TINA J YOUNG        SINDELFINGEN,GERMANY7031999021 06
77$0009.99*********
```

Running a TPG/SPG/AAS Transaction

You can use the Adapter for CICS Transactions to execute a CICS program developed for TP Gateway/Stored Procedures Gateway/Application Adapter Server (AAS).

See the *Server Release Notes* for information about migrating older TPG/SPG/AAS applications.

Chapter 16 Using the Adapter for Cloudera Impala to Access Data Managed by Hadoop

The Adapter for Cloudera Impala provides for analysis of both structured and complex data. Impala provides a JDBC driver and the Hive Query Language, a SQL-like interface with a real time query capability that shares the metadata layer and query language with Hive.

In this chapter:

Introducing the Adapter for Cloudera Impala

Hadoop is a collection of open source products and technologies administered by the Apache Software Foundation. CDH is Cloudera's Hadoop distribution. Cloudera Impala is an integrated part of CDH that provides an analytic database.

Cloudera Impala adds a real time query capability that shares the metadata layer and query language with Hive. It is available on CDH and other selected Hadoop distributions.

The Cloudera Impala adapter is JDBC based. The DataMigrator or WebFOCUS server can be run on any platform (including Windows) that connects to the server where Cloudera Impala is running.

Cloudera Impala can be accessed with either the Apache Hive JDBC driver or the Cloudera Impala JDBC driver.

Preparing the Cloudera Impala Environment

The following components are needed to use the adapter for Hadoop/Hive/Impala:

- **Java.** To use this JDBC-based adapter, you must have Java installed. Hadoop requires Version 1.7 or later. Java can be downloaded from *http://www.java.com*.

The location of Java must be specified in an environment variable.

If you are using Linux, add a line to your profile with the location where Java is installed. For example:

```
export JAVA_HOME=/usr/lib/jvm/jre-1.7.0
```

If you have JDK installed:

```
export JAVA_HOME=/usr/lib/jvm/jdk-1.7.0
```

If you are using Windows, right-click *Computer* and click *Properties*. Then click *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

```
C:\Program Files\Java\jdk7\bin\server;C:\Program Files\Java\jdk7\bin;
```

- **JDBC Driver.** The JDBC driver requires a collection of jar files. If you are using the Apache Hive JDBC driver, they are included in CDH. If you are installing the server on the same system where Hive is installed, you can point to the jar files as described in the next section. If you are installing the server on some other system, copy those files to a location of your choice. If you are using the Cloudera Impala JDBC driver, download as described below.

Procedure: How to Configure the Java CLASSPATH

The location of the JDBC driver jar files must be specified to the server. If you are running the server on the same system as the Impala server, you can specify their location. If you are running the server on another system, copy the files listed below to a location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console menu bar, click *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.
2. Expand the *Java Services* folder. Right-click *DEFAULT* and click *Properties*.

   The Java Services Configuration page opens.
3. Click the chevrons to expand Class Path.

In the IBI_CLASSPATH box, enter the full location of the Hive and Hadoop files shown below, where *hive_home* is where Hive is installed and *hadoop_home* is where Hadoop is installed. You must type them explicitly and cannot use $HIVE_HOME. The file names must be entered one per line.

If you are installing the adapter on a different system than where Hadoop and Hive are installed, copy the jar files to a location on that system.

Note: For a server running on Windows, use Windows syntax for directory names. For example:

```
C:\jdbc\hive-jdbc-1.2.0-standalone.jar
```

For the Apache Hive JDBC Driver:

```
/hive_home/lib/hive-jdbc-standalone.jar
/hadoop_home/hadoop-common.jar
```

If your Hive server is Kerberos enabled, also add:

```
/hadoop_home/client/hadoop-auth.jar
```

For Cloudera Impala

Download the JDBC driver from Cloudera. Go to the Downloads page at *http://www.cloudera.com/downloads.html*.

Click the link for Impala JDBC Driver downloads. For your operating system, select your OS Version and click the Download Now link.

This will download a file called impala_jdbc_2.5.34.zip (the numbers will change with new releases). It contains two more .zip files. Unzip the file Cloudera_Impala_JDBC41_2.5.34.zip to a location on your system (the 41 in the name indicates JDBC 4.1, which supports Java 1.7 or later). These 15 jar files constitute the Cloudera Impala JDBC driver.

Add the names of all of them to your CLASSPATH or IBI_CLASSPATH as described above.

4. Scroll down and click the *Save and Restart Java Services* button.

Configuring the Adapter for Cloudera Impala

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Cloudera Impala Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. Right-click *Cloudera Impala* and click *Configure*.

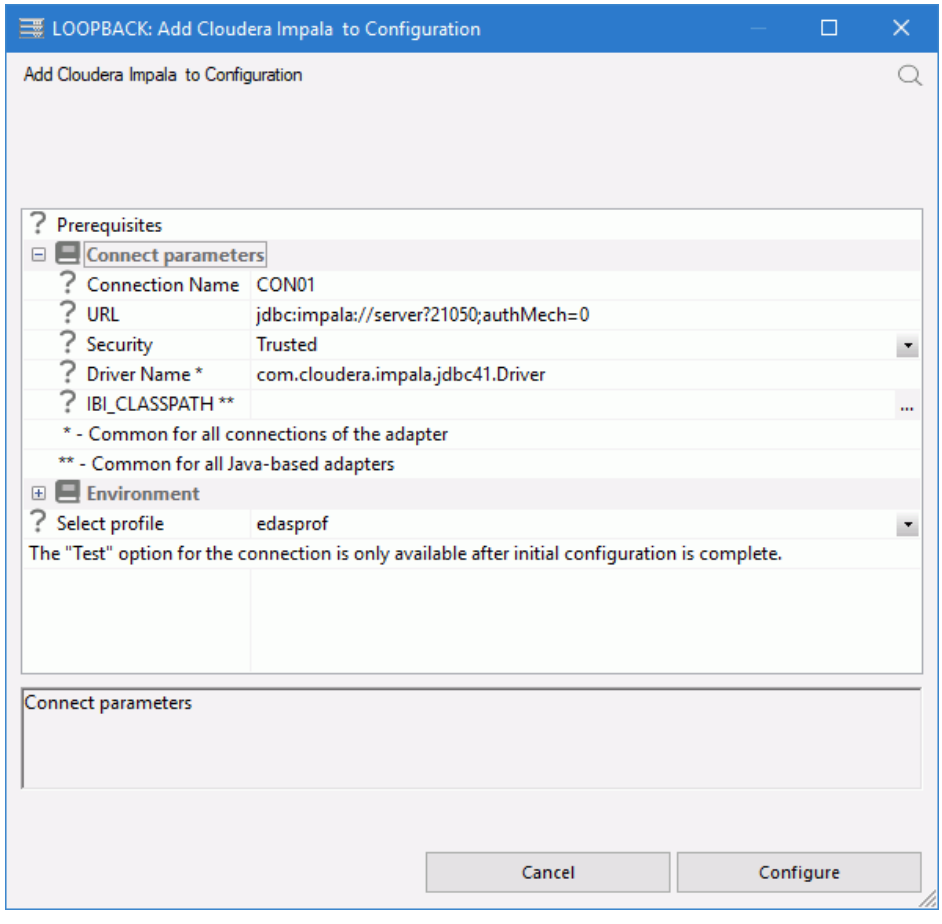
4. In the URL box, type the URL used to connect to your Hive or Impala server. For more information, see *Cloudera Impala Adapter Configuration Settings* on page 423.

5. In the Driver Name box, type the JDBC driver that you are using from the following table:

Adapter	JDBC Driver Name
Apache Hive	`org.apache.hive.jdbc.HiveDriver`
Cloudera Impala	`com.cloudera.impala.jdbc41.Driver`

6. Select the security type. If you are using Explicit, type your user ID and password.

The following image shows an example of the configuration settings used:



7. Select *edasprof* from the drop-down menu to add this connection for all users, or select a user profile.
8. Click *Test*. You should see a list of data sources on your server.
9. Click *Configure*.

Reference: Cloudera Impala Adapter Configuration Settings

The Adapter for Hadoop/Hive/Impala is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the location of the data source.

The URL used depends on which JDBC driver and which server you are connecting to.

JDBC Driver	Security	URL
Hive	None	`jdbc:hive2://server:21050/;auth=noSasl`
Hive	Kerberos (static)	`jdbc:hive2://server:21050/default; principal=impala/server@REALM.COM`
Hive	Kerberos (user)	`jdbc:hive2://server:21050/ default;principal=impala/server@REALM.COM; auth=kerberos;kerberosAuthType=fromSubject`
Impala	None	`jdbc:impala://server:21050;authMech=0`
Impala	User name and password	`jdbc:impala://server:21050; authMech=3,UID=user,PWD=pass`
Impala	Kerberos	`jdbc:impala://server:21050; AuthMech=1;KrbRealm=REALM.COM; KrbHostFQDN=server.example.com; KrbServiceName=impala`

where:

server

Is the DNS name or IP address of the system where the Hive or Impala server is running. If it is on the same system, localhost can be used.

`default`

Is the name of the default database to connect to.

`21050`

Is the default port number for an Impala server.

`auth=noSasl`

Indicates that there is no security on the Impala server.

REALM.COM

For a Kerberos enabled Impala server, this Is the name of your realm.

For further descriptions of the Impala JDBC driver options, see the *Cloudera JDBC Driver for Impala* manual.

Driver Name

Is the name of the JDBC driver, for example, org.apache.hive.jdbc.HiveDriver or com.cloudera.impala.jdbc41.Driver.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database at connection time for authentication.
- **Trusted.** The adapter connects to Impala as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and type a name in the Profile Name field (the extension is added automatically).

Kerberos

Connections to an Impala server with Kerberos enabled can be run in one of two ways:

- **Static credentials.** The same Kerberos credential is used for all connections. You must obtain a Kerberos ticket before starting the server.
- **User credentials.** Each user connecting to the server connects to Impala using credentials from the Impala Adapter connection in the server or user profile.

To set up connections to a Kerberos enabled Impala server:

1. The Reporting Server has to be secured. The server can be configured with security providers PTH, LDAP, DBMS, OPSYS, or Custom, as well as multiple security providers environment,
2. In addition to the jar files listed above, the following jar must be added to the CLASSPATH or IBI_CLASSPATH for the server:

   ```
   /hadoop_home/client/hadoop-auth.jar
   ```

Kerberos Static Requirements

In this configuration, all connections to the Impala server will be done with the same Kerberos user ID derived from the Kerberos ticket that is created before the server starts.

1. Create Kerberos ticket using:

   ```
   kinit kerbid01
   ```

   where:

   kerbid01

   Is a Kerberos ID.

2. Verify Kerberos ticket using *klist*. A message similar to the following should be returned:

```
Ticket cache: FILE:/tmp/krb5cc_532
Default  principal: kerbid01@REALM.COM

Valid Starting     Expires         Service principal
04/29/16 16:26:50  04/30/16 02:26:53  krbtgt/REALM.COM@REALM.COM
   renew until 05/06/16 16:26:50
```

3. Before configuring the Impala Adapter connection to a Kerberos enabled instance, the connection should be tested. Log in to the system running Hive and use Beeline, the native tool, to test it.
4. Start the server in the same Linux session where the Kerberos ticket was created. Log in to the Web Console and click the *Adapters* tab.
5. Right-click *Cloudera Impala* and click *Configure*. Use the following parameters to configure the adapter:

   URL

   Enter the URL. For the Apache Hive adapter, use:

   ```
   jdbc:hive2://server:21050/default;principal=hive/server@REALM.COM
   ```

   Security

   Set to *Trusted*.

6. In the Select profile drop-down menu, select the *edasprof* server profile.
7. Click *Configure*.
8. Next, configure Java services. Click the *Workspace* tab and expand the *Java Services* folder.
9. Right-click *DEFAULT* and click *Properties*.
10. Expand the *JVM Settings* section. In the JVM options box, add the following:

    ```
    -Djavax.security.auth.useSubjectCredsOnly=false
    ```

11. Restart Java services.

Once these steps are completed, the adapter can be used to access a Kerberos-enabled Hive instance.

Kerberos User Credentials Requirements

In this configuration, the connection from the server profile is used or each connected user has an Impala Adapter connection with Kerberos credentials in the user profile.

1. Enable multi-user connection processing for Kerberos by adding the following line to your profile (edasprof.prf):

   ```
   ENGINE SQLIMP SET ENABLE_KERBEROS ON
   ```

2. Configure the Impala Adapter Connection in the user profile using the following values:

   URL

   For the adapter for Apache Hive:

   ```
   jdbc:hive2://server:21050/default;principal=impala/server@REALM.COM;
   auth=kerberos;kerberosAuthType=fromSubject
   ```

   For the Cloudera Impala adapter:

   ```
   jdbc:impala://server:21050;AuthMech=1;KrbRealm=REALM.COM;
   KrbHostFQDN=server.realm.com;KrbServiceName=impala
   ```

   Security

   Set to *Explicit*

   User and Password

   Enter your Kerberos user ID and password. The server will use those credentials to create a Kerberos ticket and connect to a Kerberos-enabled Hive instance.

   Note: The user ID that you use to connect to the server does not have to be the same as the Kerberos ID you use to connect to a Kerberos enabled Hadoop cluster.

   Select Profile

   Select your profile or enter a new profile name consisting of the security provider, an underscore and the user ID. For example, *ldap01_pgmxxx*.

3. Click *Configure*.

Troubleshooting

If the server is unable to configure the connection, an error message is displayed. An example of the first line in the error message is shown below, where *nnnn* is the message number returned.

```
(FOC1400) SQLCODE IS -1 (HEX: FFFFFFFF) XOPEN: nnnn
```

Some common errors messages are:

```
[00000] JDBFOC>> connectx(): java.lang.UnsupportedClassVersionErr : or: org/
apache/hive/jdbc/HiveDriver : Unsupported major.minor version 51.0
```

The adapter requires Java 1.7 or later and your JAVA_HOME points to Java 1.6.

```
(FOC1500) ERROR: ncjInit failed. failed to connect to java server: JSS
(FOC1500) . JSCOM3 listener may be down - see edaprint.log for details
```

The server could not find Java. To see where it was looking, review the edaprint.log and set JAVA_HOME to the actual location. Finally, stop and restart your server.

```
(FOC1260)  (-1) [00000] JDBFOC>> connectx():
java.lang.ClassNotFoundException:
   org.apache.hive.jdbc.HiveDriver
(FOC1260) Check for correct JDBC driver name and environment variables.
(FOC1260) JDBC driver name is org.apache.hive.jdbc.HiveDriver
(FOC1263) THE CURRENT ENVIRONMENT VARIABLES FOR SUFFIX SQLIMP ARE :
(FOC1260) IBI_CLASSPATH : ...
 (FOC1260) CLASSPATH : C:\ibi\srv77\home\etc\flex\lib\xercesImpl.jar
```

The JDBC driver name specified cannot be found in the jar files specified in IBI_CLASSPATH or CLASSPATH. The names of the jar files are either not specified, or if specified, do not exist in that location.

```
[08S0] Could not establish connection to jdbc:hive2://hostname:21050/
auth=noSasl:
java.net.UnknownHostException: hostname
```

The server hostname could not be reached on the network. Check that the name is spelled correctly and that the system is running. Check that you can ping the server.

```
[08S01] Could not establish connection to server:21050/default:
: java.net.ConnectException: Connection refused
```

The Impala server is not listening on the specified port. Start the server if it is not running, and check that the port number is correct.

```
 (-1) [00000] JDBFOC>> makeConnection():
 javax.security.auth.login.LoginException: Pre-authentication
 information was invalid (24)
```

The Kerberos user ID and password combination is not valid.

Creating Synonyms With Cloudera Impala

Synonyms define unique names, or aliases, for each Hive table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Hive table.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Cloudera Impala

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Using Direct Pass-through With Cloudera Impala

Direct Pass-through commands, such as SHOW TABLES and DESCRIBE tablename, do not display any results.

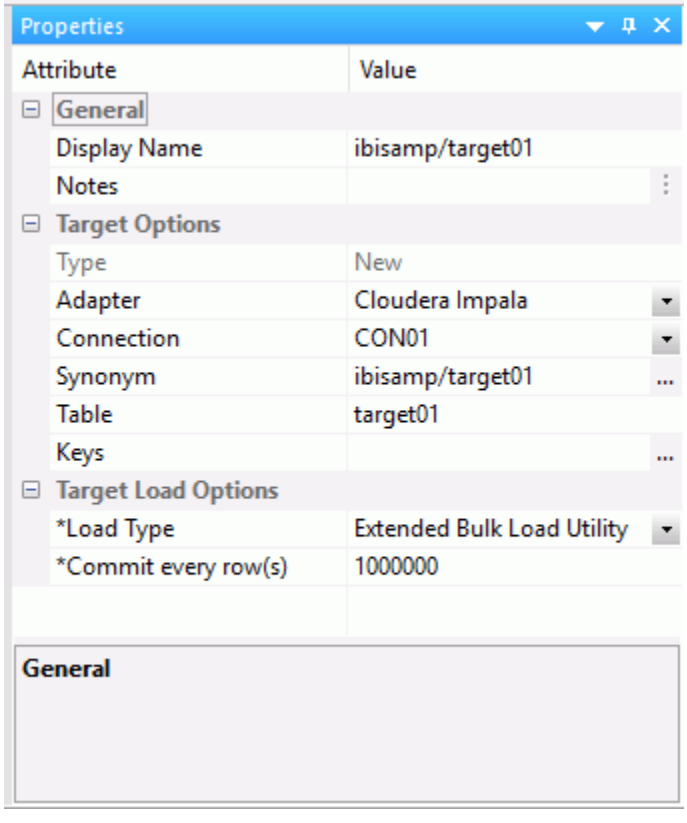
This can be avoided by adding a comment:

```
ENGINE SQLIMP
/*QUERY*/ SHOW TABLES ;
END
```

Loading Data Using DataMigrator

DataMigrator can be used to load data from any accessible data source into Hadoop and create metadata in Hive. Currently, the only load type supported is Extended Bulk Load.

To use DataMigrator to load data to Hadoop for access through Impala, we recommend that you install the DataMigrator Server on an Edge Node of your CDH cluster that has the Hadoop client software installed.If not, you can still use DataMigrator, but you must add the Fixed File adapter with the same name as the Impala adapter with an [S} FTP connection to a server where the Hadoop client is installed.



Chapter 17

Using the Adapters for C-ISAM and ISAM

The Adapters for AccuCobol C-ISAM, Informix C-ISAM, Micro Focus® C-ISAM, and FairCom c-tree ISAM allow applications to access C-ISAM and ISAM data sources. These adapters convert application requests into native C-ISAM/ISAM statements and return optimized answer sets to the requesting application.

You can also access compressed datasets with this adapter. However, the actual compression is done using the ZCOMP Exit. For details, see *Data Set Compression Exit: ZCOMP* on page 2719.

In this chapter:

Preparing the Informix C-ISAM Environment

For the Adapter for Informix C-ISAM on UNIX, no environment variable settings are required. However, if you are using the Adapter for Informix C-ISAM on Windows, you must add the location of the ISAM library called isamdll.dll (usually located in the CISAMDIR/bin directory) to the environment variable path.

For the Adapter for AccuCobol C-ISAM, you must add the location of the load libraries to the library path.

For the Adapter for Micro Focus C-ISAM, you must set and export a $COBDIR UNIX environment variable, which identifies the installation directory to the operating system.

For the Adapter for FairCom c-tree ISAM, you must add the location of the c-tree ISAM load libraries to the library path. (This adapter is only supported in the Solaris environment.)

Syntax: How to Identify the AccuCobol C-ISAM Load Library Files

```
LD_LIBRARY_PATH=path
export LD_LIBRARY_PATH
```

where:

path

Is the location of the AccuCobol C-ISAM load library files.

Note: If the server is running with security on, the IBIPATH variable needs to be used in place of LD_LIBRARY_PATH.

Syntax: How to Identify the Micro Focus C-ISAM Installation Directory

```
COBDIR=path
export COBDIR
```

where:

path

Is the location of the Micro Focus C-ISAM installation directory.

Syntax: How to Identify the FairCom c-tree ISAM Load Library Files

```
LD_LIBRARY_PATH=path
export LD_LIBRARY_PATH
```

where:

path

Is the location of the FairCom c-tree ISAM load library files.

Note: If the server is running with security on, the IBIPATH variable needs to be used in place of LD_LIBRARY_PATH.

Configuring the C-ISAM and ISAM Adapters

You can configure the adapters from the Web Console or the Data Management Console. In addition, for each type of file, you must specify the location of the data files you wish to access. You can define the location, either by:

- Including a FILEDEF command in a supported server profile. This approach enables you to quickly change the physical file name associated with the metadata.

- Specifying the location when you create a synonym. This approach enables you to define the data file location in the Master File.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Syntax: How to Specify the Location of an AccuCobol C-ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file, without the file extension.

Note:

- Read permissions must be given for the data file.
- For the Adapter for AccuCobol C-ISAM, read-only access is supported.

Syntax: How to Specify the Location of an Informix C-ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file. Omit the .dat extension.

Syntax: How to Specify the Location of a Micro Focus C-ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file.

Note:

- For each data file there must be a corresponding index file, except for Micro Focus compression 8 files, which do not have separate index files. The index is, instead, embedded in the data file.
- Read/write permissions must be given for both types of files.
- For the Informix Adapter for C-ISAM and the Micro Focus Adapter for C-ISAM, read and write access is supported.

Syntax: How to Specify the Location of a FairCom c-tree ISAM File

```
FILEDEF file_name DISK file_location
```

where:

file_name

Is the data file name.

file_location

Is the full path and file name of the data file, without the file extension.

Note:

- For each data file, there must be a corresponding index file and a corresponding parameter file.
- Read permissions must be given for all the above mentioned files.
- By default, the parameter file name is assumed to be the file_location followed by the ".p" extension.
- You can override the default parameter file name by specifying it in the environment variable EDACTPARMF.
- For the Adapter for FairCom c-tree ISAM, read-only access is supported.

Syntax: How to Specify the Location of a FairCom c-tree Parameter File

```
EDACTPARMF=path
export EDACTPARMF
```

where:

path

Is the location of the FairCom c-tree parameter file.

Note:

- By default, the parameter file name is assumed to be the file_location (as specified in a FILEDEF command or the DATASET attribute of a Master File), followed by the ".p" extension.
- You can override the default parameter file name by specifying it in the environment variable EDACTPARMF. This is required when selecting data from multiple ISAM files in a single query through a join. For such queries, a parameter file that describes all the tables involved must exist, and its name must be specified in EDACTPARMF.

Managing C-ISAM Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each C-ISAM file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for AccuCobol C-ISAM, Informix C-ISAM, Micro Focus C-ISAM, and FairCom c-tree ISAM

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Collection of COBOL definitions

Directory path

Specify the path to the directory containing multiple COBOL files.

File name

File extension

If you wish to limit retrieval, you can enter a file name and/or file extension:

- In the File name box, enter a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.
- In the File extension box, enter an extension with or without the wildcard symbol %.

Location of data files

Directory path

Enter the path to the directory containing multiple data files.

File name

File extension

If you wish to limit retrieval, you can enter a file name and/or file extension:

- In the File name box, enter a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.
- In the File extension box, enter an extension with or without the wildcard symbol %.

  Note: Filtering data files based on file extensions is not supported in the Adapter for Informix C-ISAM.

Synonym field names processing options

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize options

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Choose Data file name

After choosing tables, choose the corresponding data file name from the drop-down list:

- For AccuCobol C-ISAM, Micro Focus CISAM and Faircom c-tree ISAM, choose the file name *without* an extension.
- For Informix C-ISAM, choose the file name with the extension.*idx*.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Maintaining C-ISAM Data Sources Using SQL Commands

The Adapters for Micro Focus C-ISAM and Informix C-ISAM support SQL INSERT, UPDATE, and DELETE operations against C-ISAM data sources. This facilitates development and implementation of new applications based on C-ISAM data sources.

While processing SQL transactions, you can issue the PASSRECS command to obtain counts of rows affected by each successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax: How to Obtain the Number of Records Updated, Inserted or Deleted

```
ENGINE INT SET PASSRECS {ON|OFF}
```

where:

`INT`

Indicates that the PASSRECS setting in this command will be applied globally to all adapters that support SQL INSERT, UPDATE, and DELETE commands.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Using a Secondary Index in C-ISAM and ISAM Files

When a request contains search conditions for a field with a secondary index, the Adapters for C-ISAM and ISAM can access the secondary index directly, then retrieve the primary index value and use it to read the record. This process is much faster than reading an entire file sequentially and verifying each record against the search condition. It provides particular performance advantages when accessing large ISAM files, where secondary indexes are usually built on fields that are used in search conditions.

Example: Using a Secondary Index in a C-ISAM File

The C-ISAM file's secondary index field must be defined in the Master File with the attributes ALIAS=KEY*n* and FIELDTYPE=I.

The following Master File has a primary key field, EMPLOYEE_ID5, and four secondary indexes, SSN5, STATE_CODE5, CITIZENSHIP5 and DEPARTMENT5. These fields are defined with FIELDTYPE=I and the ALIAS names KEY1, KEY2, KEY3 and KEY4:

```
FILENAME=EMPLOYEE, SUFFIX=CISAM   ,
 DATASET=/qa/edamvt/CISAM/employee, $
  SEGMENT=SEG1, SEGTYPE=S0, $
  GROUP=G5, ALIAS=KEY, ELEMENTS=1, $
   FIELDNAME=EMPLOYEE_ID5, ALIAS=E3, USAGE=I11, ACTUAL=I4, $
   FIELDNAME=SSN5, ALIAS=KEY1, USAGE=A11, ACTUAL=A11, FIELDTYPE=I,$
   FIELDNAME=FILLER, ALIAS=E5, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=LAST_NAME5, ALIAS=E6, USAGE=A20, ACTUAL=A20, $
   FIELDNAME=FIRST_NAME5, ALIAS=E7, USAGE=A15, ACTUAL=A15, $
   FIELDNAME=FILLER, ALIAS=E8, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=BIRTHDATE5, ALIAS=E9, USAGE=I11, ACTUAL=I4, $
   FIELDNAME=SEX5, ALIAS=E10, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=FILLER, ALIAS=E11, USAGE=A3, ACTUAL=A3, $
   FIELDNAME=ETHNIC_GROU5, ALIAS=E12, USAGE=A15, ACTUAL=A15, $
   FIELDNAME=FILLER, ALIAS=E13, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=STREET5, ALIAS=E14, USAGE=A20, ACTUAL=A20, $
   FIELDNAME=CITY5, ALIAS=E15, USAGE=A15, ACTUAL=A15, $
   FIELDNAME=FILLER, ALIAS=E16, USAGE=A1, ACTUAL=A1, $

   FIELDNAME=STATE_CODE5, ALIAS=KEY2, USAGE=A2, ACTUAL=A2, FIELDTYPE=I, $
   FIELDNAME=FILLER, ALIAS=E18, USAGE=A2,ACTUAL=A2,$
   FIELDNAME=ZIP5, ALIAS=E19, USAGE=A5, ACTUAL=A5, $
   FIELDNAME=FILLER, ALIAS=E20, USAGE=A3, ACTUAL=A3, $
   FIELDNAME=HOME_PHONE5, ALIAS=E21, USAGE=A12, ACTUAL=A12, $
   FIELDNAME=WORK_PHONE5, ALIAS=E22, USAGE=A4, ACTUAL=A4, $
   FIELDNAME=CLEARANCE5, ALIAS=E23, USAGE=A2, ACTUAL=A2, $
   FIELDNAME=FILLER, ALIAS=E24, USAGE=A2, ACTUAL=A2, $
   FIELDNAME=START_DATE5, ALIAS=E25, USAGE=I11, ACTUAL=I4, $
   FIELDNAME=SALARY5, ALIAS=E26, USAGE=I11, ACTUAL=I4, $
   FIELDNAME=JOB_TITLE5, ALIAS=E27, USAGE=A24, ACTUAL=A24, $
   FIELDNAME=JOB_LEVEL5, ALIAS=E28, USAGE=I11, ACTUAL=I4, $
   FIELDNAME=SCHEDULE5, ALIAS=E29, USAGE=I11, ACTUAL=I4, $

   FIELDNAME=CITIZENSHIP5, ALIAS=KEY3, USAGE=A5, ACTUAL=A5, FIELDTYPE=I, $
   FIELDNAME=FILLER, ALIAS=E31, USAGE=A3, ACTUAL=A3, $
   FIELDNAME=SUPERVISOR5, ALIAS=E32, USAGE=P15.0, ACTUAL=P8, $

   FIELDNAME=DEPARTMENT5, ALIAS=KEY4, USAGE=I11, ACTUAL=I4, FIELDTYPE=I,  $
   FIELDNAME=DISABILITY5, ALIAS=E34, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=FILLER, ALIAS=E35, USAGE=A3, ACTUAL=A3, $
```

To process the following request,

```
TABLE FILE EMPLOYEE
PRINT EMPLOYEE_ID5 SSN5 LAST_NAME5 IF STATE_CODE5 EQ 'IL'
END
```

the adapter uses STATE_CODE5 in a secondary index search. The C-ISAM table is accessed using the E (EQ) option on the secondary key. If more than one record has the same value for the secondary key, the adapter performs a sequential read (S) on the secondary index until all values of the secondary key are retrieved.

Chapter 18

Using the Adapter for DATACOM

The Adapter for CA-DATACOM/DB allows applications to access DATACOM data sources. The adapter converts application requests into native DATACOM statements and returns optimized answer sets to the requesting application.

Note: Throughout this topic, CA-DATACOM/DB is referred to as DATACOM.

In this chapter:

Preparing the DATACOM Environment

The Adapter for CA-DATACOM/DB operates in the z/OS and USS environments and issues standard DATACOM calls for record retrieval. The adapter uses DATACOM Boolean selection capabilities, enabling it to retrieve only records that satisfy a request. This reduces the number of I/Os involved in data retrieval.

Prior to configuring the Adapter for DATACOM/DB using the Web Console, you must edit the ISTART JCL that is used to initiate the server by allocating the DATACOM.CAILIB and DATACOM.CUSLIB libraries to the server STEPLIB.

Note: If user requirement tables (URTs) are kept in a library other than CUSLIB, you may need to add an additional library to STEPLIB. For details about URTs, see *Controlling Data Access With a User Requirements Table* on page 459.

As part of your environment preparation, you must also create the AUTOURT User Table that is required to create a synonym.

Creating the AUTOURT User Table

A User Requirements Table contains all the tables that an application can access. This is the DATACOM table that is used to create a synonym. For the DATACOM data dictionary, the table is called AUTOURT. You must use the jcl that is shipped with the product to create AUTOURT.

An example of JCL that creates a AUTOURT is located in HOME.DATA in the member GENAUTO1 and GENAUTO2. For your convenience, the sample code is provided in this documentation.

Example: Sample JCL for GENAUTO1 and GENAUTO2

Run GENAUTO1 followed by GENAUTO2:

- GENAUTO1 creates the required tables and creates AUTOURT1.
- GENAUTO2 calls the intermediate file AUTOURT1 and creates the User Requirements Table, AUTOURT, which is required when you create a synonym for a DATACOM data source.

```
//GENAUTO1 JOB 'PROG',MSGCLASS=X,CLASS=S

//*************************************************************
//* NAME:     GENAUTO1
//*
//* FUNCTION: STEP1 IN CREATING DATADICTIONARY URT
//*           NEEDED FOR OUR DATACOM ADAPTER TO CREATE SYNONYMS.
//*           DO NOT CHANGE ANY OF THE ENTRIES FOR THE DBURTBL
//* SUBSTITUTIONS:-CHANGE "DATACOM.CAIMAC" TO THE NAME OF YOUR
//*                 DATACOM MACRO LIBRARY.
//*                -CHANGE  SYSLMOD TO LIBRARY WHERE YOU ARE GOING
//*                 KEEP URTS.
//*************************************************************
//ASMBL    EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR       <==CHANGE
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS,UNIT=SYSDA,
//         SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//SYSIN    DD *
         DBURSTR                                                X
               ABEND=YES,                                       X
               MULTUSE=YES,                                     X
               PRTY=7
```

```
DBURTBL                                                          X
      TBLNAM=FIL,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=FLD,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=KEY,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=REL,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=AGR,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=ALS,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES
```

```
DBURTBL                                                          X
      TBLNAM=ARA,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=ATZ,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=BAS,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=DVW,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=ELM,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=HSD,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES
```

```
DBURTBL                                                          X
      TBLNAM=JOB,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=KWC,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=LIB,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=MEM,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=MOD,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=NOD,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES
```

```
DBURTBL                                                          X
      TBLNAM=PER,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=PGM,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=PLN,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=PNL,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=PRT,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES

DBURTBL                                                          X
      TBLNAM=RPT,                                                X
      DBID=002,                                                  X
      SYNONYM=NO,                                                X
      ELMCHG=NO,                                                 X
      SEQBUFS=2,                                                 X
      UPDATE=YES
```

```
          DBURTBL                                                    X
                TBLNAM=STM,                                          X
                DBID=002,                                            X
                SYNONYM=NO,                                          X
                ELMCHG=NO,                                           X
                SEQBUFS=2,                                           X
                UPDATE=YES

          DBURTBL                                                    X
                TBLNAM=STP,                                          X
                DBID=002,                                            X
                SYNONYM=NO,                                          X
                ELMCHG=NO,                                           X
                SEQBUFS=2,                                           X
                UPDATE=YES

          DBURTBL                                                    X
                TBLNAM=SYS,                                          X
                DBID=002,                                            X
                SYNONYM=NO,                                          X
                ELMCHG=NO,                                           X
                SEQBUFS=2,                                           X
                UPDATE=YES

          DBURTBL                                                    X
                TBLNAM=TXT,                                          X
                DBID=002,                                            X
                SYNONYM=NO,                                          X
                ELMCHG=NO,                                           X
                SEQBUFS=2,                                           X
                UPDATE=YES
          DBUREND                                                    X
                USRINFO=DATA DICTIONARY URT
          END
/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NORENT,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//DBNTRY   DD DISP=(OLD,DELETE),DSN=&&OBJPASS
//SYSLMOD  DD DSN=HLQ.URTLIB,DISP=SHR              <==CHANGE
//SYSLIN   DD *
  INCLUDE DBNTRY
  NAME AUTOURT1(R)
/*
```

```
//GENAUTO2 JOB 'PROG',MSGCLASS=X,CLASS=S

//*************************************************************
//* NAME:     GENAUTO2
//*
//* FUNCTION: STEP2 IN CREATING THE DATA DICTINONARY URT
//*
//* SUBSTITUTIONS:-CHANGE "DATACOM.CAIMAC" TO THE NAME OF YOUR
//*                 DATACOM MACRO LIBRARY.
//*                -CHANGE //SYSIN TO POINT TO HOME.DATA
//*                -CHANGE  //DATAMOD TO HLQ.HOME.DATA LIBRARY
//*                -CHANGE  SYSLMOD TO LIBRARY WHERE YOU ARE GOING
//*                 KEEP URTS.
//*************************************************************

//ASMBL1   EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR          <==CHANGE
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS1,UNIT=SYSDA,
//         SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//SYSIN    DD *
         DBURINF URTABLE=LOAD,OPEN=USER,USRNTRY=USNTRY,                X
               LOADNAM=AUTOURT1
         DBUREND USRINFO=DATA DICTIONARY URT
         END

/*
//ASMBL2   EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR          <==CHANGE
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS2,UNIT=SYSDA,
//         SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//SYSIN    DD DISP=SHR,DSN=HLQ.HOME.DATA(USNTRY)   <==CHANGE
//*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NORENT,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DATAMOD  DD DISP=(OLD,DELETE),DSN=&&OBJPASS2
//DBNTRY   DD DISP=(OLD,DELETE),DSN=&&OBJPASS1
//SYSLMOD  DD DSN=HLQ.URTLIB,DISP=SHR              <==CHANGE
```

```
//SYSLIN   DD *
 INCLUDE DATAMOD
 INCLUDE DBNTRY
 ENTRY USNTRY
 NAME AUTOURT(R)
/*
```

Configuring the Adapter for DATACOM

You configure the Adapter for DATACOM from the Web Console.

Users accessing the DATACOM DBMS are authenticated by the Operating System security package, as well as any authentication performed by the DBMS itself.

Note: The Adapter for DATACOM uses the connection attributes declaration only for synonym creation purposes.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for DATACOM

The *DATACOM* adapter is under the *DBMS* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection Name

Is the logical name used to identify this particular set of attributes.

You can establish multiple connections to a Datacom data source. However, the first connection_name must define the DBID of the Datacom data dictionary, which is required as the source for Create Synonym processing.

DBID

Database ID.

For the first connection, you must specify the Data Dictionary ID.

User Table

A User Requirements Table contains all tables that an application can access. Enter a valid User Table name for access to the Datacom database with the current DBID.

For the Datacom data dictionary, this is AUTOURT.

Note that sample JCL is shipped with the product for your use in preparing the DATACOM Environment. This JCL generates the User Requirements Table, AUTOURT.

Trace

Select from the drop-down list:

CBS

Sets the accumulation of Compound Boolean Selection (CBS) diagnostics to the DATACOM diagnostics file (PXX) by making "$$$" the first 3 bytes of the User Information Block for each Datacom command.

NO

Indicates no trace. NO is the default value.

The parameter is optional.

Skipselect

Select from the drop-down list:

NO

Provides access to the root segment as stipulated in the request. This value is the default.

YES

Provides sequential access to the root segment in the Access File even if logical criteria exist for the segment.

However, if a join connection has been defined to the root segment, a selection command is used.

The parameter is optional.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE DATACOM SET CONNECTION_ATTRIBUTES connection_name /, ;
dbid: "USERTABLE urt_name [TRACE {CBS|NO}] [SKIPSELECT {YES|NO}]"
```

where:

connection_name

Is the logical name used to identify as particular set of attributes.

The connection_name that defines the DBID of the DATACOM data dictionary must be used as the source for Create Synonym processing.

dbid

Database ID.

For the first connection, you must specify the Data Dictionary ID.

urt_name

A User Requirements Table contains all tables that an application can access. Enter a valid User Table name for access to the DATACOM database with the current DBID. For related information, see *Controlling Data Access With a User Requirements Table* on page 459.

TRACE

Valid values are:

CBS sets the accumulation of Compound Boolean Selection (CBS) diagnostics to the DATACOM diagnostics file (PXX) by making "$$$" the first 3 bytes of the User Information Block for each DATACOM command.

NO indicates no trace. NO is the default value.

The parameter is optional.

SKIPSELECT

Valid values are:

NO

provides access to the root segment as stipulated in the request. This value is the default.

YES provides sequential access to the root segment in the Access File even if logical criteria exist for the segment.

However, if a join connection has been defined to the root segment, a selection command is used.

The parameter is optional.

Example: **Sample Attribute Declaration**

```
ENGINE DATACOM SET CONNECTION_ATTRIBUTES DATC /,;2: "USERTABLE AUTOURT"
ENGINE DATACOM SET CONNECTION_ATTRIBUTES ALX /,;905: "USERTABLE ALXURT"
```

Declaring Connection Attributes

Using the Default Connection

The Adapter for DATACOM uses the default connection name if the USERTABLE keyword is omitted from the Access File on the file level and the CONNECTION keyword is omitted on the segment level. For related information, see *Access Files for DATACOM* on page 475.

Syntax: How to Use the Default Connection

```
ENGINE DATACOM SET DEFAULT_CONNECTION [connection_name]
```

where:

`DATACOM`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection_name`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command.

Example: Using the Default Connection

The following SET DEFAULT_CONNECTION command selects the DATACOM database server named CON01 as the default server:

```
ENGINE DATACOM SET DEFAULT_CONNECTION CON01
```

Controlling Data Access With a User Requirements Table

First-level access to DATACOM data sources and fields is controlled by the DATACOM User Requirements Table (URT). DATACOM uses the URT to provide file and database-level security, and password protection for element-level security. The URT used for the server structure must include all the DATACOM database IDs contained in the Access File. At run time, the adapter dynamically loads the URT's load module.

The DATACOM Data Driver (like any other DATACOM application) uses a DATACOM User Requirements Table to access the appropriate DATACOM tables. A User Requirements Table contains all tables that an application can access.

One DATACOM URT containing many DATACOM tables can be described by one or many Master Files. The Access File contains the name of URT that dynamically calls the DATACOM URT.

An example of JCL that creates a DATACOM URT is located in HOME.DATA in the member GENURT1 and GENURT2. For your convenience, the sample code is provided in this documentation.

Note: To ensure the most efficient loading of URT tables, you can specify connection pooled deployment in the edaserve.cfg file from the *Workspace*, *Configuration*, *Data Services* option on the Web Console. For details, see *Configuring Deployment Modes* in *Running and Monitoring Your Server*.

Example: Sample JCL for User Requirements Table (GENURT1 and GENURT2)

Run GENURT1 followed by GENURT2.

```
//GENURT1 JOB 'PROG',MSGCLASS=X,CLASS=S
//*************************************************************
//* NAME:     GENURT1
//*
//* FUNCTION: STEP1 IN CREATING THE DATACOM URT
//*
//* SUBSTITUTIONS:-CHANGE "DATACOM.CAIMAC" TO THE NAME OF YOUR
//*                 DATACOM MACRO LIBRARY.
//*                -CHANGE MACRO DBURTBL ENTRIES:
//*                 TBLNAME   DATACOM-NAME OF TABLE  (EX PMF)
//*                 DBID      DATACOM-ID OF DATABASE (EX 001)
//*                -CHANGE MACRO DBUREND ENTRIES:
//*                 USRINFO   1-16 BYTES OF USER INFO TO BE PLACED
//*                           IN URT.
//*                -CHANGE SYSLMOD TO LIBRARY WHERE YOU ARE GOING
//*                 KEEP URTS.
//*************************************************************

//ASMBL    EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR       <==CHANGE
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS,UNIT=SYSDA,
//         SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//SYSIN    DD *
         DBURSTR MULTUSE=YES
         DBURTBL TBLNAM=XXX,DBID=XXX,SEQBUFS=2,UPDATE=YES
         DBURTBL TBLNAM=XXX,DBID=XXX,SEQBUFS=2,UPDATE=YES
         DBURTBL TBLNAM=XXX,DBID=XXX,SEQBUFS=2,UPDATE=YES
         DBUREND USRINFO=*USER COMMENTS *
         END

/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NORENT,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//DBNTRY   DD DISP=(OLD,DELETE),DSN=&&OBJPASS
//SYSLMOD DD DSN=HLQ.URTLIB,DISP=SHR            <==CHANGE
//SYSLIN    DD *
  INCLUDE DBNTRY
  NAME TESTURT1(R)
/*
//
```

```
//GENURT2 JOB 'PROG',MSGCLASS=X,CLASS=S
//*************************************************************
//* NAME:     GENURT2
//*
//* FUNCTION: STEP2 IN CREATING THE DATACOM URT
//*
//* SUBSTITUTIONS:-CHANGE "DATACOM.CAIMAC" TO THE NAME OF YOUR
//*                DATACOM MACRO LIBRARY.
//*               -CHANGE MACRO DBURINF ENTRIES:
//*                LOADNAM OBJECT MODULE CREATED FROM STEP1
//*               -CHANGE MACRO DBUREND ENTRIES:
//*                USRINFO   1-16 BYTES OF USER INFO TO BE PLACED
//*                          IN URT.
//*               -CHANGE //DATAMOD TO HLQ.HOME.DATA LIBRARY
//*               -CHANGE SYSLMOD TO LIBRARY WHERE YOU ARE GOING
//*                KEEP URTS.
//*************************************************************

//ASMBL    EXEC  PGM=ASMA90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.CAIMAC,DISP=SHR          <==CHANGE
//                DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJPASS,UNIT=SYSDA,
//                SPACE=(TRK,10),DISP=(NEW,PASS)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//SYSIN    DD *
         DBURINF URTABLE=LOAD,OPEN=USER,USRNTRY=USNTRY,                X
               LOADNAM=ALXURT1                    <==CHANGE
         DBUREND USRINFO=*USER COMMENTS*          <==CHANGE
         END

/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NORENT,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DATAMOD  DD DSN=HLQ.HOME.DATA,DISP=SHR           <==CHANGE
 //DBNTRY   DD DISP=(OLD,DELETE),DSN=&&OBJPASS
//SYSLMOD  DD DSN=PMSAEP.DATACOM.URTLIB,DISP=SHR   <==CHANGE
//SYSLIN    DD *
 INCLUDE DATMOD(USNTRY)
 INCLUDE DBNTRY
 ENTRY USNTRY
 NAME ALXURT(R)
/*
//
```

DATACOM Overview and Mapping Considerations

DATACOM is a flat file database management system. The DATACOM DATA DICTIONARY stores and manages descriptive data about a data source.

To access DATACOM data sources, you describe them to server, using server terminology and attributes. The descriptions are kept in a Master File and an associated Access File.

The Master File tells the server how to interpret the DATACOM data structure. It identifies the DATACOM:

- Elements
- Fields (for each of those elements)
- Field type and field length (as DATACOM stores them)

The Access File creates a bridge between the server and DATACOM. It translates server terminology into DATACOM syntax by identifying the appropriate DATACOM:

- Database IDs.
- 2-dimensional Tables in which all DATACOM data is stored.
- Native key fields that dictate the physical sequence in which the data is stored.
- Element security information that is used if an element is defined to DATACOM with a security code.

When the server receives a request, it:

- Looks for the Master File describing that data source.
- Finds the suffix DATACOM in the Master File. This tells the server the requested data is in a DATACOM data source.
- Loads the Adapter for DATACOM module. The adapter module then opens the Access File, in which it finds the DATACOM User Requirements Table (URT) name, the database ID(s), and file name(s).

The Adapter for DATACOM opens the URT and builds calls to DATACOM. The adapter retrieves the data and passes it to the server. The server then creates the answer set.

For details on describing data sources to a server and on Master Files and Access Files, and examples of DATACOM logical files, fields, and elements, see *Mapping Structures in the Server* on page 464.

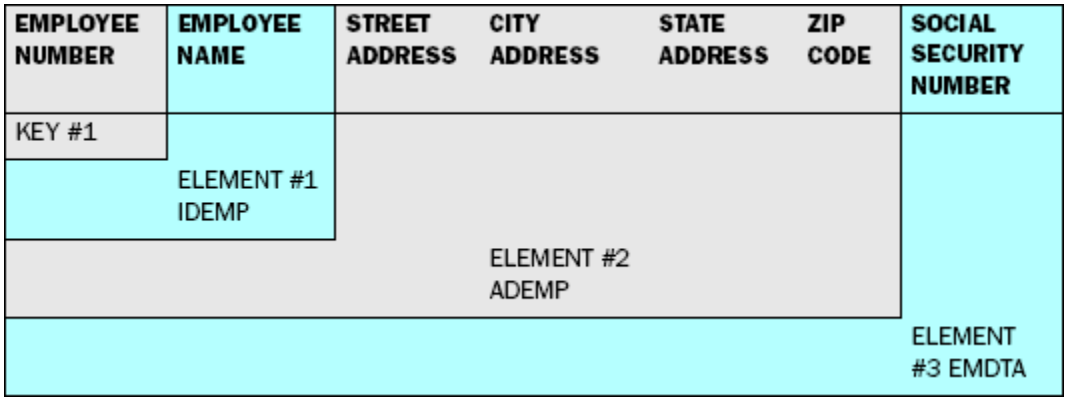
Data Structures

A DATACOM structure is a collection of one or more logical data sources, identified by a unique ID number. Each logical data source contains data records with a defined set of fields, elements, and keys, and is identified by a unique, alphanumeric 3-byte table name.

A data record consists of one or more elements. An element is the smallest logical unit of data a program can request; it has a unique 5-byte name. Elements consist of one or more contiguous fields and may have an associated security code. Because a field may belong to more than one element, the elements themselves may overlap.

DATACOM locates records through keys. Each data source may have up to 999 keys (not necessarily contiguous) consisting of 1-180 fields which total a length of 180 bytes. Each key is identified by a 5-byte name and a 3-byte numeric ID. The use of keys in the DATACOM Compound Boolean Selection Facility is transparent to the user.

The table below is the structure of the sample PERSON data source. It contains one key, three elements with fields that overlap, and seven unique fields.



Note: The DATACOM DATA DICTIONARY listing of the PERSON data source does not identify the specific fields in each element. For more information, refer to the DATA DICTIONARY Element Field Report.

The Employee Record Element (EMDTA) contains all seven fields in the data source.

The Employee Address Element (ADEMP) contains six fields that represent the employee ID and address. These are:

```
NUMBER
NAME
STREET_ADDRESS
CITY_ADDRESS
STATE_ADDRESS
ZIP_CODE_LOC
```

The Employee Identification Element (IDEMP) contains the NUMBER and NAME fields. All fields within each element are contiguous.

Mapping Structures in the Server

DATACOM terms equate to server terms as follows:

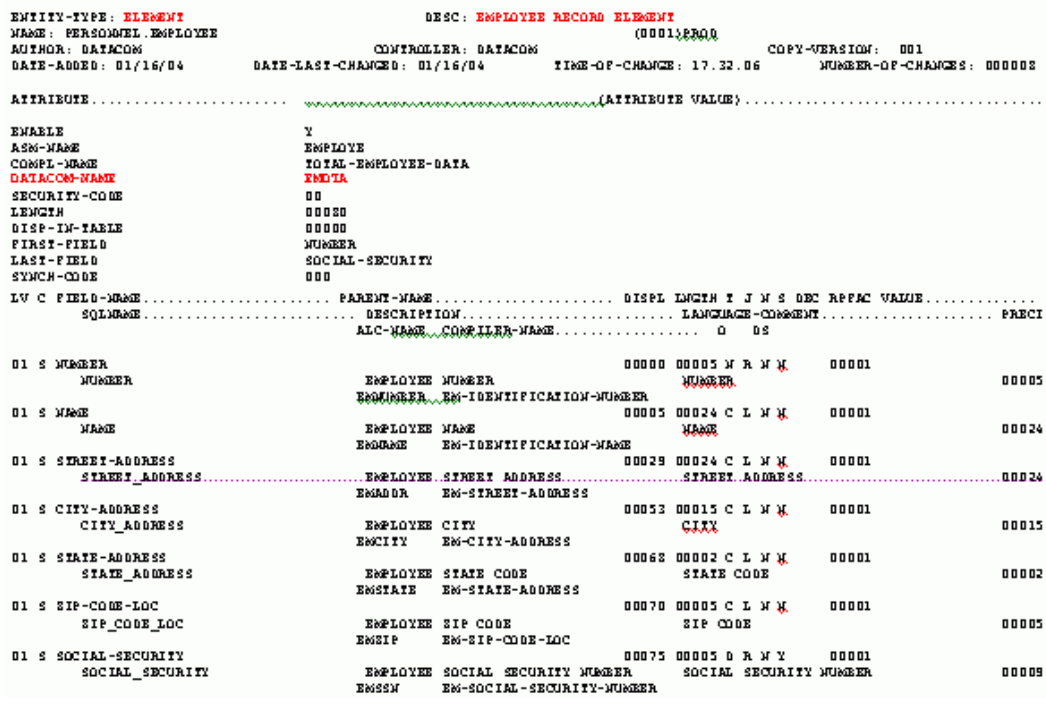
DATACOM Master File Definition	Server Master File Attributes
File	SEGMENT
Element	ALIAS for groups of fields
Field	FIELD
Field Length (LNGTH)	ACTUAL (DATACOM Format)USAGE (Server Format)

When you describe a DATACOM data source in a Master File:

- You can choose one or more elements from a DATACOM data source to build a single server segment. You do not have to specify all the elements in a particular data source.
- You must define *all fields* in the element, in the order in which they appears in the DATACOM data source. A DATACOM field becomes a server field.
- You can define two or more DATACOM elements that contain overlapping fields. However, you must define each overlapping field to the server with a unique name each time it appears in an element.
- Each DATACOM field you define to the server must have the 5-byte element name to which the DATACOM field belongs, together with a unique qualifier.
- For each DATACOM field, the DATACOM field length becomes the ACTUAL field length.
- The number of decimal positions that DATACOM indicates becomes the number of decimal positions in the USAGE format.

Example: Sample DATA DICTIONARY Element Field Report

The following example represents the DATA DICTIONARY Element Field Report for the Employee Record Element from the PERSON data source. For related information, see *Employee Record Element From PERSON Data Source* on page 465. For an explanation of the column headings in the report, see *Element Field Report Headings* on page 466.



```
ENTITY-TYPE: ELEMENT                          DESC: EMPLOYEE RECORD ELEMENT
NAME: PERSONNEL.EMPLOYEE                                    (0001)PROD
AUTHOR: DATACOM                        CONTROLLER: DATACOM                  COPY-VERSION:   001
DATE-ADDED: 01/16/04       DATE-LAST-CHANGED: 01/16/04       TIME-OF-CHANGE: 17.32.06      NUMBER-OF-CHANGES: 000008

ATTRIBUTE.....................  ..............................(ATTRIBUTE VALUE).....................................

ENABLE                         Y
ASM-NAME                       EMPLOYE
COMPL-NAME                     TOTAL-EMPLOYEE-DATA
DATACOM-NAME                   EMDTA
SECURITY-CODE                  00
LENGTH                         00020
DISP-IN-TABLE                  00000
FIRST-FIELD                    NUMBER
LAST-FIELD                     SOCIAL-SECURITY
SYNCH-CODE                     000

LV C FIELD-NAME..................... PARENT-NAME..................... DISPL LNGTH T J M S DEC RPFAC VALUE.............
       SQLNAME....................... DESCRIPTION....................... LANGUAGE-COMMENT..................... PRECI
                                  ALC-NAME  COMPILER-NAME.................  O   DS

01 S NUMBER                                                          00000 00005 N R M M     00001
       NUMBER                       EMPLOYEE NUMBER                      NUMBER                          00005
                                    EMNUMBER  EM-IDENTIFICATION-NUMBER
01 S NAME                                                            00005 00024 C L M M     00001
       NAME                         EMPLOYEE NAME                        NAME                            00024
                                    EMNAME    EM-IDENTIFICATION-NAME
01 S STREET-ADDRESS                                                  00029 00024 C L M M     00001
       STREET_ADDRESS...............EMPLOYEE STREET ADDRESS.............STREET_ADDRESS.....................00024
                                    EMADDR    EM-STREET-ADDRESS
01 S CITY-ADDRESS                                                    00053 00015 C L M M     00001
       CITY_ADDRESS                 EMPLOYEE CITY                        CITY                            00015
                                    EMCITY    EM-CITY-ADDRESS
01 S STATE-ADDRESS                                                   00068 00002 C L M M     00001
       STATE_ADDRESS                EMPLOYEE STATE CODE                  STATE CODE                      00002
                                    EMSTATE   EM-STATE-ADDRESS
01 S ZIP-CODE-LOC                                                    00070 00005 C L M M     00001
       ZIP_CODE_LOC                 EMPLOYEE ZIP CODE                    ZIP CODE                        00005
                                    EMZIP     EM-ZIP-CODE-LOC
01 S SOCIAL-SECURITY                                                 00075 00005 D R M Y     00001
       SOCIAL_SECURITY              EMPLOYEE SOCIAL SECURITY NUMBER      SOCIAL SECURITY NUMBER          00009
                                    EMSSN     EM-SOCIAL-SECURITY-NUMBER
```

Example: Employee Record Element From PERSON Data Source

The following Master File and Access File describe the Employee Record Element from the DATACOM PERSON data source.

Master File PERSON

```
FILENAME=PERSON, SUFFIX=DATACOM , $
  SEGMENT=PERSON, SEGTYPE=S0, $
    FIELDNAME=NUMBER,          ALIAS=EMDTA.0000, USAGE=P5,  ACTUAL=Z5,  $
    FIELDNAME=NAME,            ALIAS=EMDTA.0001, USAGE=A24, ACTUAL=A24, $
    FIELDNAME=STREET_ADDRESS,  ALIAS=EMDTA.0002, USAGE=A24, ACTUAL=A24, $
    FIELDNAME=CITY_ADDRESS,    ALIAS=EMDTA.0003, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=STATE_ADDRESS,   ALIAS=EMDTA.0004, USAGE=A2,  ACTUAL=A2,  $
    FIELDNAME=ZIP_CODE_LOC,    ALIAS=EMDTA.0005, USAGE=A5,  ACTUAL=A5,  $
    FIELDNAME=SOCIAL_SECURITY, ALIAS=EMDTA.0006, USAGE=P10, ACTUAL=P5,  $
```

Access File PERSON

```
USERTABLE=SAMPURT, TRACE=NO, $
 SEGNAME= PERSON, TABLENAME=PMF, DBID=1, KEYNAME=EMPNO, $
  FIELD=NUMBER, SIGN=N, TYPE=C, $
  FIELD=SOCIAL_SECURITY, SIGN=Y, TYPE=C, $
```

Reference: Element Field Report Headings

Column Heading	Definition
LV (LEVEL)	The two-digit field level number.
C (CLASS)	Code representing the field class. Codes for the valid classes are: C = Compound (or parent) field F = Filler field I = Index S = Simple field V = Value (or qualifier) field
FIELD-NAME	Data dictionary occurrence name of the field.
PARENT-NAME (RECORD-NAME)	Data dictionary occurrence name of the parent field. Applies to compound fields only.
DISPL(DISP-IN-TABLE)	Displacement of the field in the record or table.
LNGTH (LENGTH)	Length of the field in bytes. For fields defined with a V data type (VARCHAR), this value includes the automatically generated two-byte field that defines the data's length.

Column Heading	Definition
T (TYPE)	Code representing the data type of the field. Codes for the valid data types are: B = Binary C = Character D = Packed decimal E = Extended-precision floating-point G = Graphics character set H = Hexadecimal with 2 characters per entry K = Kanji L = Long-precision floating-point N = Numeric field zoned decimal S = Short-precision floating-point T = PL/1 bit representation V = Variable length character Y = Pure DBCS Double-Byte Character Set Z = Mixed DBCS and EBCDIC 2 = Halfword binary 4 = Fullword binary 8 = Double word binary
J (JUSTIFICATION)	Justification: L = Left-justified R = Right-justified
N (NULL-INDICATOR)	Null indicator: N = No null-indicator, a one-byte indicator preceding this field. Y = Null indicator is present.
S (SIGN)	Sign: Y = Signed N = No sign
DEC (DECIMALS)	Decimal places for numeric, packed decimal, and binary fields.
RPFAC (REPEAT)	Repetition factor.

Column Heading	Definition
VALUE	Value attribute. This is either the default value of the field or the value associated with a qualified (CLASS=V) field.
DESCRIPTION	FIELD entity-occurrence description.
PRECISION	Specifies the maximum number of displayable digits allowed for data stored in fields. Available TYPEs are: D = Decimal N = Numeric G = Graphic V = Varchar W = Vargraphic

Managing DATACOM Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the DATACOM data types.

For the Adapter for DATACOM to access DATACOM files, you must describe each file you use in a Master and Access File. The logical description of an DATACOM file is stored in a Master File, which describes the field layout. The physical attributes of the DATACOM file, such as the database number and security are stored in the Access File.

Creating Synonyms

Synonyms define unique names (or aliases) for each DATACOM table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for DATACOM

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Filter Data Base Name

The first Create Synonym pane displays the connection name, user table, and data dictionary ID that were supplied during configuration. To see all related databases, simply click *Next*.

If you wish to filter the list of available databases, click the *Filter Data Base* check box. A Name input box is added to the pane.

Type a string for filtering the list of databases, inserting the wildcard character (%) as needed at the beginning and/or end of the string, then click *Next*.

For example, enter: ABC% to select all databases whose names begin with the letters ABC; %ABC to select databases whose names end with the letters ABC; %ABC% to select databases whose names contain the letters ABC at the beginning, middle, or end.

Data Base Name

On the second Create Synonym pane, select a database from the drop-down list and click *Next*.

Filter Table Name

If you wish to filter the list of available tables, click the *Filter Table Name* check box. A Name input box is added to the pane.

Type a string for filtering the list of tables, inserting the wildcard character (%) as needed at the beginning and/or end of the string, then click *Next*.

Table Name

On the third Create Synonym pane, select a table in the selected database from the drop-down list and click *Next*.

URT or Connection

On the final Create Synonym pane, indicate whether you want to generate the synonym with a user requirements table (URT) or a connection name. The generated *Access File Attributes* on page 480 depend on the method you choose here. (For an illustration of the differences, see *Access Files Generated by URT and Connection* on page 485.)

If you choose URT, enter the name of a user requirements table. This table defines the database and files that a user can access.

Note that this entry is not required for creating a synonym, however, data cannot be accessed without it. For more information about the user requirements table, see *Controlling Data Access With a User Requirements Table* on page 459.

If you choose Connection, enter the name of the connection that defines the URT and Database ID for accessing Datacom data.

Synonym name

Displays the name that will be assigned to the synonyms. To assign a different name, replace the displayed value.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Select elements

An element is the smallest unit of data. An element contains fields that can belong to more than one element.

- To select all elements in the list, select the check box to the left of the *Element Name* column heading.
- To select specific elements, select the corresponding check boxes.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Master Files for DATACOM

There are three types of Master File declarations.

Each declaration must begin on a separate line. A declaration consists of attribute-value pairs separated by commas. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.

Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

File Attributes

Each Master File begins with a file declaration. The file declaration has two attributes, FILENAME and SUFFIX

```
FILE[NAME]=filename, SUFFIX=DATACOM [,$]
```

where:

filename

Identifies the Master File. The file name can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents.

DATACOM

Is the value for the Adapter for CA-DATACOM/DB.

Segment Attributes

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes, SEGNAME and SEGTYPE

```
SEGNAME=segname, SEGTYPE=type [,$]
```

where:

segname

Is the segment name that serves as a link to the actual DATACOM table name. It can consist of a maximum of 8 alphanumeric characters. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name.

The SEGNAME value in the Master File must be identical to the SEGNAME value specified in the Access File.

type

Indicates the type of segment sequencing:

SO indicates the segments are logically sequenced in low-to-high order.

U indicates a unique segment.

Field Attributes

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the following primary field attributes:

FIELDNAME

Identifies the name of a field.

ALIAS

Identifies the DATACOM 5-byte element short name to which the field belongs, along with a unique qualifier.

USAGE

Identifies how to display a field on reports.

ACTUAL

Identifies the datatype and length in bytes for a field.

You can get values for these attributes from the DATACOM DATA DICTIONARY Element Field Report.

The syntax for a field declaration is

```
FIELD[NAME]=fieldname, ALIAS=elementname.qualifier,
 [USAGE=]display_format, [ACTUAL=]storage_format ,$
```

where:

fieldname

Is the unqualified name of the field. You must describe the fields in the order in which they appear in the DATACOM element. You can find the DATACOM field names, their relative position within the element, and the DATACOM formats in the DATA DICTIONARY Element Field Report. This value must be unique within the Master File. The name can consist of a maximum of 66 alphanumeric characters. The name must begin with a letter. Special characters and embedded blanks are not recommended.

Note that some elements contain overlapping fields. You must define overlapping fields for those elements in which they appear, and assign a unique field name each time.

Note: You must describe all the columns of the DATACOM element in your Master File. However, if there is a column(s) you do not need, you can supply filler information as a placeholder.

elementname

Is the DATACOM 5-byte element short name, followed by a period, to which the field belongs. You can find the name in the DATACOM DATA DICTIONARY listing of Master Files.

qualifier

Is the qualifier that is separated from the element name by a period. The qualifier is used to make the ALIAS unique and provide the field with an alternate identification. The total length of the ALIAS, including the element name and qualifier, cannot exceed 12 characters.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column *includes* the decimal point and negative sign. In SQL, the total length of the field or column *excludes* the decimal point and negative sign.

For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

For related information, see *USAGE Conversion Formats From DATACOM* on page 475.

storage_format

Is the storage format of the DATACOM data type and length, in bytes, for the field. You can find this under LNGTH in the field report.

Reference: **USAGE Conversion Formats From DATACOM**

The following table provides information about conversion formats for DATACOM data types.

DATACOM Type	DATACOM Length	USAGE Type	USAGE Length
C (Character)	*n*	A (Alphanumeric)	*n*
N (Numeric; Zoned decimal)		P (Packed decimal)	*n* +1 (if decimal) +1 (if sign space)
D (Packed Decimal)	*n*	P (Packed decimal)	(2**n*)-1 (+ 1 if decimal) +1 (if sign space)
B (Binary)	2	I (Integer)	4
	4	I (Integer)	10

where:

n

Is the DATACOM length (LNGTH) of the fields.

Access Files for DATACOM

Each Master File must have a corresponding Access File. The file name of the Access File must be the same as that used for the Master File.

The Access File serves as a link between the server and DATACOM. It stores such information as the URT name, DATACOM Database ID(s), DATACOM table name(s), keyname(s), and element security information. In addition, the Access File provides you with the TRACE parameter, a useful debugging tool.

The Access File contains four types of logical records:

- Header record
- Segment record
- Element record
- Field record

Each consists of a list of keyword and value pairs, separated by commas and terminated by a comma and dollar sign (,$). The list is free-form and may span several lines; keywords may be in any order.

Syntax: How to Use the Header Logical Record

The header record contains these attributes

```
[USERTABLE=urt_name
 [,TRACE={NO|CBS}]
 [,SEQACCESS=[NO|YES}]
 [,FOUNDIMIT=nnnnnn]
 [,REJECTLIMIT=nnnnnn]
,$]
```

where:

urt_name

Is the name of the load module that supplies the DATACOM User Requirements Table (URT) information to be dynamically loaded. The URT must contain all the database IDs and file specified in the Master File, or you will not have access to the records.

This attribute is generated when you create a synonym based in a specified URT.

TRACE

NO indicates no trace. NO is the default value.

CBS sets the accumulation of Compound Boolean Selection (CBS) diagnostics to PXX by making "$$$" the first 3 bytes of the User Information Block for each Datacom command.

This attribute is generated when you create a synonym based on the specified URT.

SEQACCESS

NO provides access to the root segment as stipulated in the request. This value is the default.

Note: While SEQACCESS is a valid keyword, it is supported for backward compatibility and must be added to the Access File *manually*. You can perform the same function in the connection string using the SKIPSELECT option.

YES provides sequential access to the root segment in the Access File even if logical criteria exist for the segment.

However, if a join connection has been defined to the root segment, a selection command is used.

FOUNDLIMIT/REJECTLIMIT

Enable you to set interrupt limits to monitor the process of record selection. Accepted (Found) or Rejected records are physical records that either satisfy or do not satisfy the provided search values for non-indexed fields.

nnnnnn

Is the number of Interrupt Records rejected or accepted (found). When this number is reached, selection terminates. This value is applied for *all segments* of the Access File.

If required, you can add these attributes to the Access File manually.

Syntax: How to Use the Segment Logical Record

The segment record contains these attributes

```
SEGNAME=segname
 ,{DBID=dbid|CONNECTION=connection_name}
 ,TABLENAME=tablename
 ,KEYNAME=keyname
[,KEYFLD=keyfld, IXFLD=ixfld,]
[,FOUNDLIMIT=nnnnnn]
[,REJECTLIMIT=nnnnnn]
,$
```

where:

segname

Identifies the segment name. The Master File SEGMENT attribute is the same as the SEGNAME attribute in the Access File.

dbid

Identifies the numeric DATACOM database ID.

This attribute is generated when you create a synonym based on a specified URT.

connection_name

May be used in the Access file on the segment level. The value of this keyword is the valid connection _name that was previously set by the SET CONNECTION_ATTRIBUTES command.

Connection is used in the Access File only when the file level keyword USERTABLE is omitted. Otherwise, connection information is ignored.

Values for the keywords USERTABLE, DBID, TRACE, and SEQACCESS (or SKIPSELECT) are taken from the connection string and used for the current segment only.

This attribute is generated when you create a synonym based on a specified connection.

tablename

Identifies the 3-byte DATACOM table name to which the segment identified by SEGNAME corresponds. You can find this name in the DATA DICTIONARY Element Field Report, under DBNAME.

keyname

Is the 5-character alphanumeric name of the key being used in sequential access commands (GSETL/GETIT). The key name is defined in the Data dictionary KEY entity-occurrence field DATACOM-NAME.

For fastest sequential retrieval, specify the Native Key, which is in the DATA DICTIONARY Indented Report, where MN stands for mandatory native. If the KEYNAME value is not provided, records are processed in the physical sequence using the GSETP/GETPS commands. If a key other than the Native Key is specified, the data is internally requested. (Note that Random retrieval is slower than sequential retrieval.)

Note: KEYNAME does not have to be the native key. If you wish to use a different keyname, that is acceptable. However, be aware that you may not get all the records.

keyfld

Is the server field name from the parent segment of a cross-reference; it is mandatory for everything but the root segment. You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value must be contiguous, and on the same line. The key list can span multiple lines.

If required, you can add this attribute to the Access File *manually*.

ixfld

Is the server field name in the cross-referenced segment that establishes the cross-reference. It is mandatory for all but the root segment. The value(s) of these field(s) must have USAGE and ACTUAL formats comparable to the KEYFLD.

You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value may not span more than one line. The key list can span multiple lines.

If required, you can add this attribute to the Access File *manually*.

`FOUNDLIMIT/REJECTLIMIT`

Enable you to set interrupt limits to monitor the process of record selection. Accepted (Found) or Rejected records are physical records that either satisfy or do not satisfy the provided search values for non-indexed fields.

nnnnnn

Is the number of Interrupt Records rejected or accepted (found). When this number is reached, selection terminates. This value is applied *only to the designated segment*.

If required, you can add these attributes to the Access File manually.

Syntax: How to Use the Element Logical Record

You must specify an element logical record if an element is defined to DATACOM with a security code.

The element record contains these attributes

`ELEMENTNAME=`*`element`*`, SECURITY=`*`hh`*` ,$`

where:

element

The 5-byte name of the element.

hh

The two-position hexadecimal value of the 1-byte DATACOM security code.

If the Data Dictionary contains a value for the element, this attribute is automatically generated in the Access File when you create a synonym.

Syntax: How to Use the Field Logical Record

The field record contains these attributes

`FIELD=`*`fieldname`*`, SIGN={Y|N}, TYPE={C|P|U|Q} ,$`

where:

SIGN

Has the same value as the option SIGN associated with this field in the Data Dictionary. Valid values are:

Y indicates signed fields. This is the default value.

N indicates unsigned fields.

TYPE

Has the same value as the option TYPE-NUMERIC associated with this field in the Data Dictionary. The synonym is automatically generated with the correct numeric type:

C to indicate a conventional signed field (either positive or negative). Data must have the sign C for a positive signed fields and D for a negative signed fields.

P to indicate a conventional signed positive field. Data must have the sign C.

U to indicate an unconventional signed field (either positive or negative). Data may have the unconventional signs A, B, or E. This is the default value.

Q to indicate an unconventional signed positive field. Data may have the unconventional signs A or E.

Note: This attribute is generated when you create a synonym based on the specified URT or Connection.

Reference: Access File Attributes

The following chart lists Access File attributes generated during synonym creation. Note that attributes vary depending on whether the synonym is generated based on a user requirements table or a connection string.

Keyword	Description
CONNECTION	May be used in the Access file on the segment level. The value of this keyword is the valid connection _name that was previously set by the SET CONNECTION_ATTRIBUTES command. Connection is used in the Access File only when the file level keyword USERTABLE is omitted. Otherwise, connection information is ignored. Values for the keywords USERTABLE, DBID, TRACE, and SEQACCESS (or SKIPSELELCT) are taken from the connection string and used for the current segment only. **Note:** This attribute is generated when you create a synonym based on a specified connection. For related syntax, see *How to Use the Segment Logical Record* on page 477.
DBID	Identifies the numeric, 5-byte DATACOM database ID. It can be 5 digits with a maximum of 5000 values. Values for DBID are taken from the connection string. They will be used for the current segment only. **Note:** This attribute is generated when you create a synonym based on a specified URT. For related syntax, see *How to Use the Segment Logical Record* on page 477.
ELEMENTNAME SECURITY	You must specify an element logical record if an element is defined to DATACOM with a security code. If the Data Dictionary contains a value for the element, this attribute is automatically generated in the Access File when you create a synonym. For related syntax, see *How to Use the Element Logical Record* on page 479.
FIELD	You must specify a field logical record if a file has a numeric (N) or decimal (D) Datacom data type. For related syntax, see *How to Use the Field Logical Record* on page 479.

Keyword	Description
FOUNDLIMIT REJECTLIMIT	You can set interrupt limits to monitor the process of record selection. Accepted (Found) or Rejected records are physical records that either satisfy or do not satisfy the provided search values for non-indexed fields. If required, you can add these attributes to the Access File *manually*. For related syntax, see *How to Use the Header Logical Record* on page 476 and *How to Use the Segment Logical Record* on page 477.
IXFIELD	Is the server field name in the cross-referenced segment that establishes the cross-reference. It is mandatory for all but the root segment. The value(s) of these field(s) must have USAGE and ACTUAL formats comparable to the KEYFLD. You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value may not span more than one line. The key list can span multiple lines. If required, you can add this attribute to the Access File *manually*. For related syntax, see *How to Use the Segment Logical Record* on page 477.

Keyword	Description
KEYNAME	Is a 5-character alphanumeric name of the key being used in the sequential access commands (GSETL and GETIT). The key name is defined in the Data Dictionary KEY entity-occurrence field, DATACOM-NAME. For fastest sequential retrieval, specify the Native Key, which is in the Data Dictionary Indented Report, where MN stands for mandatory native. If the KEYNAME value is not provided, records are processed in the physical sequence using the commands GSETP/GETPS commands. If a key other than the Native key is specified, the data is internally requested. (Note that Random retrieval is slower than sequential retrieval.) KEYNAME does not have to be the native key. You may assign a different keyname, however, if you do so you may not get all the records. For related syntax, see *How to Use the Segment Logical Record* on page 477.
KEYFIELD	Is the server field name from the parent segment of a cross-reference; it is mandatory for everything but the root segment. You may also specify multiple field names, concatenated with the forward slash symbol (/) without blanks. The keyword value must be contiguous, and on the same line. The key list can span multiple lines. If required, you can add this attribute to the Access File *manually*. For related syntax, see *How to Use the Segment Logical Record* on page 477.
SEGNAME	Identifies the segment name. The Master File SEGMENT attribute is the same as the SEGNAME attribute in the Access File. For related syntax, see *How to Use the Segment Logical Record* on page 477.

Keyword	Description
SEQACCESS	While SEQACCESS is a valid keyword, it is supported for backward compatibility and must be added to the Access File *manually*. You can perform the same function in the connection string using the SKIPSELECT option. Values for SEQACCESS are taken from the connection string. They will be used for the current segment only. For related syntax, see *How to Use the Header Logical Record* on page 476.
TRACE	Enables you to set the accumulation of Compound Boolean Selection (CBS) diagnostics to the DATACOM diagnostics file (PXX) by making "$$ $" the first 3 bytes of the User Information Block for each Datacom command. **Note:** This attribute is generated when you create a synonym based on the specified URT. For related syntax, see *How to Use the Header Logical Record* on page 476.
TABLENAME	Identifies the 3-byte DATACOM table name to which the segment identified by SEGNAME corresponds. You can find this name in the DATA DICTIONARY Element Field Report, under DBNAME. For related syntax, see *How to Use the Segment Logical Record* on page 477.

Keyword	Description
USERTABLE	Identifies the load module that supplies the DATACOM User Requirements Table (URT) information to be dynamically loaded. The URT must contain all the database IDs and file specified in the Master File, or you will not have access to the records. Values for the keyword USERTABLE are taken from the connection string. They will be used for the current segment only. **Note:** This attribute is generated when you create a synonym based in a specified URT. For related syntax, see *How to Use the Header Logical Record* on page 476.

Example: **Access Files Generated by URT and Connection**

This Access File was generated with the URT option on the Create Synonym pane.

```
USERTABLE=ALXURT, TRACE=NO, $
 SEGNAME=ORDERS, TABLENAME=ORD, DBID=10, KEYNAME=ORDID, $
  FIELD=ORD_ID,       SIGN=N, TYPE=C, $
  FIELD=ORD_ID1,      SIGN=N, TYPE=C, $
  FIELD=ORD_ID2,      SIGN=N, TYPE=C, $
  FIELD=DISC_PCT,     SIGN=Y, TYPE=C, $
  FIELD=ORDER_TOTAL, SIGN=Y, TYPE=C, $
  FIELD=FRT_AMT,      SIGN=Y, TYPE=C, $
  FIELD=ORD_AMT,      SIGN=Y, TYPE=C, $
```

This Access file was generated with the Connection option on the Create Synonym pane.

```
SEGNAME=TABLEA, TABLENAME=CAT, CONNECTION=CON02, KEYNAME=NATKY, $
  ELEMENTNAME=ALEX1, SECURITY=e1, $
  FIELD=READING1, SIGN=N, TYPE=C, $
  FIELD=PATMRN,    SIGN=N, TYPE=C, $
  ELEMENTNAME=ALEX2, SECURITY=e2, $
  ELEMENTNAME=ALEX3, SECURITY=e3, $
```

Describing Multi-File Structures for DATACOM

You can describe many different DATACOM data sources in one Master File and Access File pair. However, in the adapter, there must be at least one field in common between any parent and descendant pair of data sources. Each set of related fields must also have comparable USAGE and ACTUAL formats.

Each time you add a descendant segment, you must specify the PARENT attribute in the segment record. This will identify hierarchical relationships between the data sources.

In addition, you must specify the relationship between the fields in the Access File with the KEYFLD and IXFLD attributes:

- The value of the KEYFLD can be a simple field, a DEFINE field, or a list of fields.
- The value of IXFLD can be a simple field, or a list of fields.
- The length, and the format, of KEYFLD and IXFLD must be the same.

There are advantages to defining multiple data sources in a single structure:

- The multi-file structure creates a view of the DATACOM data source.
- You can describe up to 64 separate but related DATACOM logical files as segments in a single Master File. This will allow you to issue a request from any or all of the 64 segments defined in a single Master File without issuing a JOIN command.

To illustrate this concept, we will use the DATACOM data sources PERSON and PAYROLL.

The PAYROLL data source contains information about the wages, commissions, and taxes for each person in the PERSON data source.

Master File PAYROLL

```
FILENAME=PAYROLL, SUFFIX=DATACOM , $
  SEGMENT=PAYROLL, SEGTYPE=S0, $
   FIELDNAME=NUMBER,          ALIAS=PAYRC.0000, USAGE=P5,   ACTUAL=Z5, $
   FIELDNAME=ACTIVITY_CODE,   ALIAS=PAYRC.0001, USAGE=A1,   ACTUAL=A1, $
   FIELDNAME=ACTIVITY_STATUS, ALIAS=PAYRC.0002, USAGE=A1,   ACTUAL=A1, $
   FIELDNAME=CURRENT_RATE,    ALIAS=PAYRC.0003, USAGE=P9.2, ACTUAL=Z8, $
   FIELDNAME=YTD_WAGES,       ALIAS=PAYRC.0004, USAGE=P9.2, ACTUAL=Z8, $
   FIELDNAME=YTD_COMMISSION,  ALIAS=PAYRC.0005, USAGE=P9.2, ACTUAL=Z8, $
   FIELDNAME=YTD_TAX,         ALIAS=PAYRC.0006, USAGE=P9.2, ACTUAL=Z8, $
```

Access File PAYROLL

```
USERTABLE=SAMPURT, TRACE=NO, $
 SEGNAME=PAYROLL, TABLENAME=PAY, DBID=1, KEYNAME=EMPNO, $
  FIELD=NUMBER,          SIGN=N, TYPE=C, $
  FIELD=CURRENT_RATE,    SIGN=N, TYPE=C, $
  FIELD=YTD_WAGES,       SIGN=N, TYPE=C, $
  FIELD=YTD_COMMISSION, SIGN=N, TYPE=C, $
  FIELD=YTD_TAX,         SIGN=N, TYPE=C, $
```

Multi-File Master File

Consider an application that contains both the PERSON and PAYROLL data sources. It is reasonable to generate reports that identify the current salary rate for all employees, or a list of all the employees who have a particular salary rate.

For demonstration purposes, we will define both these DATACOM logical files in one Master File. In all but the top (or root) segment, you must specify the PARENT attribute. PARENT establishes the relationship between two segments

```
PARENT=segname
```

where:

segname

Is the name of the parent of the child segment. If you do not specify a PARENT attribute, it will default to the immediate predecessor of the child segment as the parent.

Example: Multi-File Master File

The following example shows a static Join in a manually combined multi-file Master File with the PARENT attribute added to the descendent segment. The file name PERSPAY identifies the entire Master File.

Master File PERSPAY

```
FILENAME=PERSPAY, SUFFIX=DATACOM , $
 SEGMENT=PERSON, SEGTYPE=S0, $
  FIELDNAME=NUMBER_PERS,     ALIAS=EMDTA.0000, USAGE=P5,   ACTUAL=Z5,  $
  FIELDNAME=NAME,            ALIAS=EMDTA.0001, USAGE=A24,  ACTUAL=A24, $
  FIELDNAME=STREET_ADDRESS,  ALIAS=EMDTA.0002, USAGE=A24,  ACTUAL=A24, $
  FIELDNAME=CITY_ADDRESS,    ALIAS=EMDTA.0003, USAGE=A15,  ACTUAL=A15, $
  FIELDNAME=STATE_ADDRESS,   ALIAS=EMDTA.0004, USAGE=A2,   ACTUAL=A2,  $
  FIELDNAME=ZIP_CODE_LOC,    ALIAS=EMDTA.0005, USAGE=A5,   ACTUAL=A5,  $
  FIELDNAME=SOCIAL_SECURITY, ALIAS=EMDTA.0006, USAGE=P10,  ACTUAL=P5,  $
 SEGMENT=PAYROLL, SEGTYPE=S0, PARENT=PERSON, $
  FIELDNAME=NUMBER_PAY,      ALIAS=PAYRC.0000, USAGE=P5,   ACTUAL=Z5,  $
  FIELDNAME=ACTIVITY_CODE,   ALIAS=PAYRC.0001, USAGE=A1,   ACTUAL=A1,  $
  FIELDNAME=ACTIVITY_STATUS, ALIAS=PAYRC.0002, USAGE=A1,   ACTUAL=A1,  $
  FIELDNAME=CURRENT_RATE,    ALIAS=PAYRC.0003, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_WAGES,       ALIAS=PAYRC.0004, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_COMMISSION,  ALIAS=PAYRC.0005, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_TAX,         ALIAS=PAYRC.0006, USAGE=P9.2, ACTUAL=Z8,  $
```

Multi-File Access File

The Access File for a multi-file structure must have a segment record for each logical segment you define in the Master File.

Each segment record, other than the root, must contain the KEYFLD and IXFLD attributes in addition to the Access File attributes described earlier. These new attributes identify the field(s) in an embedded cross-reference. The common field(s) serve as the cross-referenced link between the two data sources.

The PAYROLL data source is the child, or cross-referenced, segment in the following example. This is important because you identify cross-referencing fields by specifying KEYFLD and IXFLD in the cross-referenced segment record.

KEYFLD

The field name in the parent or host segment, or a list of up to five fields separated by a forward slash symbol (/).

IXFLD

The field name in the child or cross-referenced segment, or a list of up to five fields separated by a forward slash symbol (/).

Example: Multi-File Access File

In the example, the field NUMBER_PERS in the PERSON data source is related to the field NUMBER_PAY in the PAYROLL data source. Both have the same USAGE and ACTUAL data format and length (P5 and Z5); this is a server requirement.

Add the segment record for the PAYROLL data source to the Access File, including the KEYFLD and IXFLD attributes:

Access File PERSPAY

```
USERTABLE=SAMPURT, TRACE=NO, $
 SEGNAME=PERSON, TABLENAME=PMF, DBID=1, KEYNAME=EMPNO, $
  FIELD=NUMBER, SIGN=N, TYPE=C, $
  FIELD=SOCIAL_SECURITY, SIGN=Y, TYPE=C, $
 SEGNAME=PAYROLL, TABLENAME=PAY, DBID=1, KEYNAME=EMPNO,
          KEYFLD=NUMBER_PERS, IXFLD=NUMBER_PAY, $
  FIELD=NUMBER,          SIGN=N, TYPE=C, $
  FIELD=CURRENT_RATE,    SIGN=N, TYPE=C, $
  FIELD=YTD_WAGES,       SIGN=N, TYPE=C, $
  FIELD=YTD_COMMISSION, SIGN=N, TYPE=C, $
  FIELD=YTD_TAX,         SIGN=N, TYPE=C, $
```

Using Multiple Fields to Cross-Reference Data Sources

With the Adapter for DATACOM, you can use up to five fields to establish a relationship or cross-reference between data sources.

For example, matching permutations of values of Z/Y/X/W/V in the KEYFLD and IXFLD attributes can be used to create new cross-referenced fields in the Access File. This is sometimes referred to as using concatenated keys.

Example: Using Multiple Fields to Cross-Reference Data Sources

To illustrate how to use concatenated keys, we will alter the PERSPAY multi-file Master and Access Files as shown in the following example:

1. Replace the field NUMBER_PERS in the PERSON segment with the LAST_NAME and FIRST_NAME fields.
2. In the PAYROLL segment, replace the field NUMBER_PAY with LN and FN.
3. Separate each field that is part of the cross-reference with a forward slash symbol (/).

```
FILENAME=PERPAY, SUFFIX=DATACOM , $
 SEGMENT=PERSON, SEGTYPE=S0, $
  FIELDNAME=LAST_NAME,       ALIAS=EMDTA.0000, USAGE=A15,  ACTUAL=A15, $
  FIELDNAME=FIRST_NAME,      ALIAS=EMDTA.0001, USAGE=A10,  ACTUAL=A10, $
  FIELDNAME=NAME,            ALIAS=EMDTA.0002, USAGE=A24,  ACTUAL=A24, $
  FIELDNAME=STREET_ADDRESS,  ALIAS=EMDTA.0003, USAGE=A24,  ACTUAL=A24, $
  FIELDNAME=CITY_ADDRESS,    ALIAS=EMDTA.0004, USAGE=A15,  ACTUAL=A15, $
  FIELDNAME=STATE_ADDRESS,   ALIAS=EMDTA.0005, USAGE=A2,   ACTUAL=A2,  $
  FIELDNAME=ZIP_CODE_LOC,    ALIAS=EMDTA.0006, USAGE=A5,   ACTUAL=A5,  $
  FIELDNAME=SOCIAL_SECURITY, ALIAS=EMDTA.0007, USAGE=P10,  ACTUAL=P5,  $
 SEGMENT=PAYROLL, SEGTYPE=S0, PARENT=PERSON, $
  FIELDNAME=LN,              ALIAS=EMDTA.0000, USAGE=A15,  ACTUAL=A15, $
  FIELDNAME=FN,              ALIAS=EMDTA.0001, USAGE=A10,  ACTUAL=A10, $
  FIELDNAME=ACTIVITY_CODE,   ALIAS=PAYRC.0002, USAGE=A1,   ACTUAL=A1,  $
  FIELDNAME=ACTIVITY_STATUS, ALIAS=PAYRC.0003, USAGE=A1,   ACTUAL=A1,  $
  FIELDNAME=CURRENT_RATE,    ALIAS=PAYRC.0004, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_WAGES,       ALIAS=PAYRC.0005, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_COMMISSION,  ALIAS=PAYRC.0006, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_TAX,         ALIAS=PAYRC.0007, USAGE=P9.2, ACTUAL=Z8,  $
```

These multiple fields will now become the KEYFLD and IXFLD attributes in the Access File.

Separate each field from the next with the forward slash symbol (/), and add these attributes to the PAYROLL segment record in the Access File:

Access File PERPAY

```
USERTABLE=SAMPURT, TRACE=NO, $
 SEGNAME=PERSON, TABLENAME=PMF, DBID=1, KEYNAME=EMPNO, $
  FIELD=SOCIAL_SECURITY, SIGN=Y, TYPE=C, $
 SEGNAME=PAYROLL, TABLENAME=PAY, DBID=1, KEYNAME=EMPNO,
         KEYFLD=LAST_NAME/FIRST_NAME, IXFLD=LN/FN, $
  FIELD=CURRENT_RATE,    SIGN=N, TYPE=C, $
  FIELD=YTD_WAGES,       SIGN=N, TYPE=C, $
  FIELD=YTD_COMMISSION, SIGN=N, TYPE=C, $
  FIELD=YTD_TAX,         SIGN=N, TYPE=C, $
```

Reviewing a DATACOM Detail Report

Although the synonym creation facility generates metadata for a DATACOM data source, you may find it useful to review internal information, as it is maintained by DATACOM, in a detail report like the sample shown in this section.

Example: **Detailed DATACOM Field Report**

This is a sample DATACOM field report. (Note that non-essential information has been removed from this multi-page report, as indicted by ellipses, to focus your attention on the basic content.)

```
Date: 01/25/2006 Time: 11.43.12      ***********************************************************         Page:        1
                                     *  Advantage CA-Datacom Database Datadictionary Option  *
Security Level: 1                    *                    Utility Report                       *         Version: 11.0
                                     * Copyright 2003 Computer Associates International, Inc.*
User: DATACOM-INSTALL                ***********************************************************         DD Base:     2

-RPT START,TABLE,PERSONNEL,STANDARD                                   ;
-RPT DETAIL,TABLE                                                     ;
-RPT DETAIL,ELEMENT                                                   ;
-RPT FIELD,ELEMENT                                                    ;
-END                                                                  ;

ENTITY-TYPE: TABLE                          DESC: PERSONNEL MASTER FILE
NAME: PERSONNEL                                                     (0001)PROD
AUTHOR: DATACOM                     CONTROLLER: DATACOM                          COPY-VERSION:  001
DATE-ADDED: 01/16/04       DATE-LAST-CHANGED: 01/16/04       TIME-OF-CHANGE: 17.32.06       NUMBER-OF-CHANGES: 000016

ATTRIBUTE.......................  (ATTRIBUTE VALUE)..................................

ENABLE                            Y
LENGTH                            00080
FILENAME                          PERSONNEL
AUTHID                            SYSUSR
SQLNAME                           PERSONNEL
SQL-ACCESS                        Y
CONSTRAINT                        N
FIPS                              N
SQL-SECURITY                      N
DBMS-USED                         DATACOM
SQL-INTENT                        Y
SQL-CATAL-DATE                    01/16/04
SQL-CATAL-TIME
DATACOM-NAME                      PMF
DATACOM-ID                        002
DUPE-MASTER-KEY                   Y
CHNG-MASTER-KEY                   Y
LOGGING                           Y
RECOVERY                          Y
COMPRESSION                       Y
PIPELINE-OPTION                   Y
COMPRESS-EXIT
ENCRYPTION-KEY
CLUSTER-KEY-LEN                   000
CLUSTER-KEY-ID                    000
MAX-RECORD-SIZE                   00080
MONITOR-NM
DD-ENTITY-TABLE                   N
ENTY-NAME-LEN                     00
ENTY-HIST-VERS                    000
BASE-NAME                         HUMAN-RESOURCE
BASE-VER                          00001
DATABASE-ID                       00001
DD-MASTER-KEY                     NUMBER
DB-MASTER-KEY                     EMPNO
DD-NATIVE-KEY                     NUMBER
DB-NATIVE-KEY                     EMPNO
DL1T-CONSTRAINT                   N
DDD-SYNCH                         Y
CNS-UNIQUE                        N
CNS-DOMAIN                        N
CNS-REFERENCED                    N
CNS-REFERS                        N
CNS-SAME-BASE                     Y
TABLEKIND
FALLBACK
PARTITION                         N
ANY-PARTITION-AUTHID
ANY-PARTITION-SQLNAME
ANY-PARTITION-DATACOM-NAME
MOVE-ROW-TO-NEW-PARTITION         N
SEGMENT-NM
INIT-PARM
COMPRESS-RTN
```

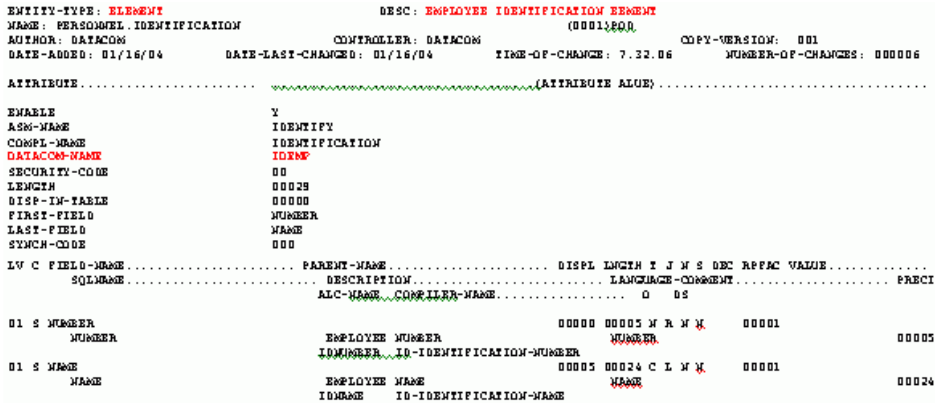
...

```
ENTITY-TYPE: ELEMENT                          DESC: EMPLOYEE RECORD ELEMENT
NAME: PERSONNEL.EMPLOYEE                                          (0001)PROD
AUTHOR: DATACOM                          CONTROLLER: DATACOM              COPY-VERSION:  001
DATE-ADDED: 01/16/04        DATE-LAST-CHANGED: 01/16/04       TIME-OF-CHANGE: 17.32.06      NUMBER-OF-CHANGES: 000008

ATTRIBUTE.....................  vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv(ATTRIBUTE VALUE).....................................

ENABLE                         Y
ASM-NAME                       EMPLOYE
COMPL-NAME                     TOTAL-EMPLOYEE-DATA
DATACOM-NAME                   EMDTA
SECURITY-CODE                  00
LENGTH                         00080
DISP-IN-TABLE                  00000
FIRST-FIELD                    NUMBER
LAST-FIELD                     SOCIAL-SECURITY
SYNCH-CODE                     000

LV C FIELD-NAME..................... PARENT-NAME..................... DISPL LNGTH T J M S DEC RPFAC VALUE.............
       SQLNAME....................... DESCRIPTION..................... LANGUAGE-COMMENT..................... PRECI
                                     ALC-NAME  COMPILER-NAME.................  0   DS

01 S NUMBER                                                          00000 00005 N R M N     00001
       NUMBER                         EMPLOYEE NUMBER                    NUMBER                          00005
                                     EMNUMBER  EM-IDENTIFICATION-NUMBER
01 S NAME                                                            00005 00024 C L M N     00001
       NAME                           EMPLOYEE NAME                      NAME                            00024
                                     EMNAME    EM-IDENTIFICATION-NAME
01 S STREET-ADDRESS                                                  00029 00024 C L M N     00001
       STREET_ADDRESS.................EMPLOYEE STREET ADDRESS.....STREET ADDRESS.....00024
                                     EMADDR    EM-STREET-ADDRESS
01 S CITY-ADDRESS                                                    00053 00015 C L M N     00001
       CITY_ADDRESS                   EMPLOYEE CITY                      CITY                            00015
                                     EMCITY    EM-CITY-ADDRESS
01 S STATE-ADDRESS                                                   00068 00002 C L M N     00001
       STATE_ADDRESS                  EMPLOYEE STATE CODE                STATE CODE                      00002
                                     EMSTATE   EM-STATE-ADDRESS
01 S ZIP-CODE-LOC                                                    00070 00005 C L M N     00001
       ZIP_CODE_LOC                   EMPLOYEE ZIP CODE                  ZIP CODE                        00005
                                     EMZIP     EM-ZIP-CODE-LOC
01 S SOCIAL-SECURITY                                                 00075 00005 U R M Y     00001
       SOCIAL_SECURITY                EMPLOYEE SOCIAL SECURITY NUMBER    SOCIAL SECURITY NUMBER          00009
                                     EMSSN     EM-SOCIAL-SECURITY-NUMBER
```

...

```
ENTITY-TYPE: ELEMENT                          DESC: EMPLOYEE ADDRESS ELEMENT
NAME: PERSONNEL.FULL-ADDRESS                                      (0001)PROD
AUTHOR: DATACOM                          CONTROLLER: DATACOM              COPY-VERSION:  001
DATE-ADDED: 01/16/04        DATE-LAST-CHANGED: 01/16/04       TIME-OF-CHANGE: 17.32.06      NUMBER-OF-CHANGES: 000007

ATTRIBUTE.....................  vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv(ATTRIBUTE VALUE).....................................

ENABLE                         Y
ASM-NAME                       FULLADDR
COMPL-NAME                     FULL-EMPLOYEE-ADDRESS
DATACOM-NAME                   ADDMP
SECURITY-CODE                  00
LENGTH                         00075
DISP-IN-TABLE                  00000
FIRST-FIELD                    NUMBER
LAST-FIELD                     ZIP-CODE-LOC
SYNCH-CODE                     000

LV C FIELD-NAME..................... PARENT-NAME..................... DISPL LNGTH T J M S DEC RPFAC VALUE.............
       SQLNAME....................... DESCRIPTION..................... LANGUAGE-COMMENT..................... PRECI
                                     ALC-NAME  COMPILER-NAME.................  0   DS

01 S NUMBER                                                          00000 00005 N R M N     00001
       NUMBER                         EMPLOYEE NUMBER                    NUMBER                          00005
                                     ADNUMBER  AD-NUMBER
01 S NAME                                                            00005 00024 C L M N     00001
       NAME                           EMPLOYEE NAME                      NME                             00024
                                     ADNAME    AD-IDENTIFICATION-NAME
01 S STREET-ADDRESS                                                  00029 00024 C L M N     00001
       STREET_ADDRESS.................EMPLOYEE STREET ADDRESS.....STREET ADDRESS.....00024
                                     ADSTREET  AD-STREET-ADDRESS
01 S CITY-ADDRESS                                                    00053 00015 C L M N     00001
       CITY_ADDRESS                   EMPLOYEE CITY                      CITY                            00015
                                     ADCITY    AD-CITY-ADDRESS
01 S STATE-ADDRESS                                                   00068 00002 C L M N     00001
       STATE_ADDRESS                  EMPLOYEE STATE CODE                SATE CODE                       00002
                                     ADSTATE   AD-STATE-ADDRESS
01 S ZIP-CODE-LOC                                                    00070 00005 C L M N     00001
       ZIP_CODE_LOC                   EMPLOYEE ZIP CODE                  ZP CODE                         00005
                                     ADZIP     AD-ZIP-CODE-LOCATION
```

...



Data Retrieval Logic for DATACOM

This section describes the three types of DATACOM record retrieval commands generated by the Adapter for DATACOM:

- **Sequential Logical Retrieval Commands:** *GSETL and GETIT*. These commands are issued by the adapter only for the root of the accessed subtree, and only if the request does not specify any record selection criteria on that segment in the Access File and the value of the parameter KEYNAME is provided.
- **Sequential Physical Retrieval Commands:** *GSETP and GETPS*. These commands are issued by the adapter only for the root of the accessed subtree, and only if the request does not specify any record selection criteria on that segment in the Access File and the value of the parameter KEYNAME is not provided.
- **Compound Boolean Selection Commands:** *SELFR and SELNR*. These commands are issued for DATACOM records when there are available selection criteria.

Note: Sequential retrieval commands are used under each of the following circumstances:

- The parameter SEQACCESS=YES is specified in the Access File for the root segment.
- The SET SKIPSELECT =YES command is issued.
- The Skipselect option is specified in the connection string.

GSETL and GETIT

When the adapter determines sequential retrieval needs to be done for the root segment, it issues a GSETL command. It uses the KEYNAME, specified in the Access File, and establishes the starting position at the beginning (lowest value) of the key. Only one GSETL command is issued per request.

GETIT commands follow the GSETL command. They retrieve the element(s) for each root record indexed by the key. The adapter requests only the elements necessary to satisfy the request. If there are no sort fields in the request, the answer set is produced, in ascending order, by the key. GETIT commands are issued repeatedly until DATACOM issues a return code of 19 (End of Table).

As long as the KEYNAME in the Access File is the Native Key, the adapter retrieves all records that correspond to a DATACOM table.

GSETP and GETPS

Only one GSETP command is issued per request. GETPS commands follow the GSETP command. They retrieve the element(s) for each root record by physical layout in the database. GETPS commands are issued repeatedly until DATACOM issues a return code of 19 (End of Table).

SELFR and SELNR

The adapter uses two types of selection criteria to construct the SELFR call to DATACOM for a record:

1. All the values supplied in WHERE statements that mention a field in the segment and have any of these types of operators:

```
EQ          GT          NE

IS          LE          CONTAINS

GE          LT          FROM...TO
```

   Note: The SELFR request does not use selection on defined fields.

2. DBA value selection criteria on the segment.

The adapter generates a SELFR command, using all available selection criteria on the segment. This builds a list of records which match the selection criteria and returns the first record. The list is built in the temporary index area of the DATACOM data source.

For a descendant record with SEGTYPE=U, the SELFR retrieves the unique descendant. No SELNR command is issued. Otherwise, the SELFR command is followed by SELNR command(s) which retrieve the records listed in the temporary index. A SELNR is issued repeatedly until DATACOM issues a return code of 14 (End/Beginning of Set).

Note: An additional selection command, SELST, enables you to interrupt record selection if records are rejected from the index based on the REJECTLIMIT attribute in the Access File. For related information, see FOUNDLIMIT and REJECTLIMIT in *Access File Attributes* on page 480.

Example: Using a Sequence of Calls for a Multi-Segment Request

To illustrate the sequence of calls the adapter must make to DATACOM to service a multi-segment request, we will use the PERSPAY data source.

Master File PERSPAY

```
FILENAME=PERSPAY, SUFFIX=DATACOM , $
 SEGMENT=PERSON, SEGTYPE=S0, $
  FIELDNAME=NUMBER_PERS,      ALIAS=EMDTA.0000, USAGE=P5,    ACTUAL=Z5, $
  FIELDNAME=NAME,             ALIAS=EMDTA.0001, USAGE=A24,   ACTUAL=A24, $
  FIELDNAME=STREET_ADDRESS,   ALIAS=EMDTA.0002, USAGE=A24,   ACTUAL=A24, $
  FIELDNAME=CITY_ADDRESS,     ALIAS=EMDTA.0003, USAGE=A15,   ACTUAL=A15, $
  FIELDNAME=STATE_ADDRESS,    ALIAS=EMDTA.0004, USAGE=A2,    ACTUAL=A2,  $
  FIELDNAME=ZIP_CODE_LOC,     ALIAS=EMDTA.0005, USAGE=A5,    ACTUAL=A5,  $
  FIELDNAME=SOCIAL_SECURITY, ALIAS=EMDTA.0006, USAGE=P10,   ACTUAL=P5,  $
 SEGMENT=PAYROLL, SEGTYPE=S0, PARENT=PERSON, $
  FIELDNAME=NUMBER_PAY,       ALIAS=PAYRC.0000, USAGE=P5,    ACTUAL=Z5, $
  FIELDNAME=ACTIVITY_CODE,    ALIAS=PAYRC.0001, USAGE=A1,    ACTUAL=A1,  $
  FIELDNAME=ACTIVITY_STATUS, ALIAS=PAYRC.0002, USAGE=A1,    ACTUAL=A1,  $
  FIELDNAME=CURRENT_RATE,     ALIAS=PAYRC.0003, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_WAGES,        ALIAS=PAYRC.0004, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_COMMISSION,  ALIAS=PAYRC.0005, USAGE=P9.2, ACTUAL=Z8,  $
  FIELDNAME=YTD_TAX,          ALIAS=PAYRC.0006, USAGE=P9.2, ACTUAL=Z8,  $
```

In the following request

```
SELECT NAME, NUMBER_PERS, SOCIAL_SECURITY, YTD_WAGES, YTD_COMMISSION,
   YTD_TAX
FROM PERSPAY
WHERE ACTIVITY_STATUS = 'A';
```

PERSON is a referenced segment; it is the root of the accessed subtree. NAME, NUMBER_PERS, and SOCIAL_SECURITY are the names of fields on this segment.

However, there are no available selection criteria for the PERSON segment (ACTIVITY_STATUS is a field on the PAY segment). Therefore, the adapter will first issue a GSETL command for the PAYROLL MASTER File (PMF), using the NUMBER_PERS Native Key. The GSETL command is followed by a GETIT command to retrieve the first root record, the EMDTA element.

Since the server always processes from top-to-bottom, left-to-right, all the related descendants of this first root record must be retrieved before proceeding to the next root record.

Next, to generate the SELFR call to retrieve PAY data source records related to the PERSON parent, two pieces of selection criteria will be used: the value of NUMBER_PERS (the KEYFLD) from the PMF record, and the selection criteria on ACTIVITY_STATUS.

The first SELFR call to DATACOM for the first root record will retrieve a PAY record with NUMBER_PAY equal to NUMBER_PERS, and ACTIVITY_STATUS equal to 'A'. Subsequent SELNR calls (SEGTYPE=SO) may retrieve other records with those same values.

After DATACOM returns a 14 for the PAY data source, a second GETIT command will be generated for a PMF record. The value of this record's NUMBER_PERS field is used, with ACTIVITY_STATUS EQ A, to generate a new set of SELFR/SELNR calls to retrieve related PAY records.

This process will be repeated until DATACOM returns a 19 for the GETIT command on the root, signifying that all records have been retrieved and processed. If the request were

```
SELECT NAME NUMBER_PERS SOCIAL_SECURITY YTD_WAGES YTD_COMMISSION YTD_TAX
FROM PERSPAY
WHERE ACTIVITY_STATUS = 'A' AND
STATE_ADDRESS IN ('CT', 'RI', 'MA', 'VT', 'NH', 'ME';
```

SELFR/SELNR commands would be issued instead of GSETL/GETIT commands, with the five STATE_ADDRESS values (STATE_ADDRESS is a field on the PERSON segment). The process however, would be the same.

Finally, keep in mind:

- The adapter will retrieve all descendant records of one parent occurrence before it will retrieve the next parent record.
- The higher you put your selection criteria in the hierarchical structure, the more efficient processing will be.

Chapter 19

Using the Adapter for Db2

The Adapter for Db2 allows applications to access Db2 data sources. The adapter converts data or application requests into native Db2 statements and returns optimized answer sets to the requesting program.

The adapter supports the execution of Db2 stored procedures and Db2 Cube Views.

In this chapter:

Preparing the Db2 Environment

The CLI version of the Adapter for Db2 requires the installation of the Db2 Client. The CAF version requires Db2 load libraries. The JDBC version requires jar files to be installed. All versions of the adapter allow connection to local or remote Db2 servers.

Procedure: How to Set Up the Environment on Microsoft Windows

Two versions of the Adapter for Db2 are available in this environment. Each one requires different pre-configuration steps prior to using the Web Console.

Db2 CLI:

The adapter supports Unicode data. You must add DB2CODEPAGE=1208 to the Windows environment either by:

- Setting the environment variable in Windows.

- Setting the environment variable in the edaenv.cfg file.

```
DB2CODEPAGE=1208
```

Db2 JDBC:

Identify the location of the JDBC Driver files by adding them to the environment variable CLASSPATH before server startup. For example, set:

```
CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
```

Procedure: How to Set Up the Environment on UNIX

Two versions of the Adapter for Db2 are available in this environment. Each one requires different pre-configuration steps prior to using the Web Console.

Db2 CLI:

- Specify the Db2 instance to access using the UNIX environment variable $DB2INSTANCE. For example:

  ```
  export DB2INSTANCE=db2
  ```

- Specify the location of the Db2 instance you wish to access using the UNIX environment variable $INSTHOME. For example:

  ```
  export INSTHOME=/usr/db2
  ```

- Specify the path to the Db2 shared library using the UNIX environment variable $LD_LIBRARY_PATH. For example:

  ```
  export LD_LIBRARY_PATH=$INSTHOME/sqllib/lib:$LD_LIBRARY_PATH
  ```

  Note that if the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

- **Unicode support.** The adapter supports Unicode data. You must set the LANG environment variable in the edastart file or in a separate shell file. For example, for American English, you would export the following variable:

  ```
  export LANG=EN_US.UTF-8
  ```

  To take advantage of adapter support for Unicode data, the Reporting Server must be configured with code page 65001.

Db2 JDBC:

Identify the location of the JDBC Driver files by adding them to the environment variable CLASSPATH before server startup. For example, issue the following commands:

```
CLASSPATH=/usr/driver_files/mydriver.jar;
$CLASSPATH
export CLASSPATH
...
```

Procedure: How to Set Up the Environment on z/OS

Two deployment options are supported on this platform that support the CLI and CAF versions of the adapter. Each one requires different pre-configuration steps prior to using the Web Console.

- **Db2/CAF:** Utilizes embedded SQL using the IBM/Db2 for z/OS Call Attach Facility to access the Db2 RDBMS.
  - **For HFS deployment.** The Web Console configuration panel for the bind operation has three options (Yes, Create JCL only, and No). Select *Yes* to execute a shell script ($EDAHOME/bin/genidb2.sh) to bind your plan to Db2. If you do not have bind authority over the Db2 subsystem, then select *Create JCL* only so that your Db2 DBA can run member GENDB2 in the configuration data set for you. Select *No* if you do not want to run the bind script or create the JCL. The Adapter for Db2 can now be used.
  - **For PDS deployment.** The Web Console configuration panel for the bind operation will only create the JCL member GENDB2 in the configuration data set. This job will have to be submitted by your Db2 DBA to bind the plan to Db2 before the Db2 adapter can be used.

  For the naming convention for the configuration data set, see the installation manual for z/OS.
- **Db2/CLI:** Utilizes IBM/Db2 for z/OS Call Level Interface calls to access the Db2 RDBMS.

  The adapter makes use of the Db2 plan DSNACLI. To ensure that all users of the adapter have access to this plan, issue the following command in the Db2 environment:

  ```
  GRANT EXECUTE ON PLAN DSNACLI TO PUBLIC
  ```

  The adapter also uses the DSNAOINI environment variable or DSNAOINI DDNAME pointing to the Db2 CLI initialization file. You can provide this variable at installation time using the ISETUP panels, or you can add the variable manually to the EDAENV member of the installation data set after completing the installation.

The following is an example for the DSNAOINI environment variable.

```
DSNAOINI=full_path_name/db2cli.ini
```

The following is an example for the DSNAOINI DDNAME.

```
//DSNAOINI DD DSN=library(member),DISP=SHR
```

or

```
//DSNAOINI DD DSN=dataset,DISP=SHR
```

Unicode support. The adapter supports Unicode data in Db2 databases that have been created with the CCSID UNICODE option. You must ensure that the DSNAOINI environment variable or DDNAME points to a configuration file containing the following specification:

```
CURRENTAPPENSCH=UNICODE
```

Procedure: How to Access a Remote Database Server on IBM i

Using the standard rules for deploying the Db2 CLI Client, the server supports connections to:

- Local Db2 servers.
- Remote Db2 servers.

When using Db2/CLI to connect to a remote Db2 data source, the Db2 client catalog must contain an entry for the node where the remote data source resides and an entry for the remote database name.

Support for Remote Aliases. An alias entry in a local Db2 system that points to a table in another Db2 system is only supported with the CREATE SYNONYM command. It is not supported for metadata queries when SET ENGINE=DB2 is in effect.

Configuring the Adapter for Db2

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to the Db2 database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Db2 database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Db2 Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   The Adapters folder opens.
2. Expand the *Available* folder, if it is not already expanded or click +.
3. Expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the connection attributes reference.

6. Click *Configure* or *Add*. The configured adapter is added to the Adapters list in the navigation pane.

Reference: Connection Attributes for Db2 With CLI

The Db2 adapter in under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource/DSN

Db2 data source name (DSN), is a locally registered alias of a remote DB2 database. There is no default data source name. You must enter a value.

For IBM i, this is the Remote Database Directory entry or *LOCAL (for local host).

For z/OS, this is the Db2 location name as specified in the Db2 communications data source.

DSN-less

Applies to Linux, UNIX, and Windows. Check this box to specify the following connection parameters instead of a DSN. Note that bulk load is not supported with this type of connection.

Host Name

Is the name of the host where the Db2 server is running.

Port Number

Is the port number on which the Db2 server is listening.

Database Name

Is the Db2 database name.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Trusted Context

Select this check box to enable a Db2 trusted context, which is a database object that defines a trust relationship for a connection between the database and an external application server. The trust relationship is based upon the attributes defined in the trusted context object.

If you select Enable, the following is added after the password in the SET CONNECTION ATTRIBUTES command:

```
:'trusted_context=y'
```

For example, assume the Reporting server is configured with LDAP security and the WebFOCUS Client is configured with SECURITY TRUSTED. From the WebFOCUS Client, the Web Console is opened without an authentication prompt and connection to the server is trusted. The adapter connects to the Db2 database with the user ID in the connection string. It checks for a trusted context object and then switches users.

Additional connection string keywords (Optional)

You can enter any additional connection string keywords separated by semicolons (;). Each keyword consists of an attribute and value pair in the form attribute=value.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Reference: Connection Attributes for Db2 With CAF

This list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

SSID

Db2 SSID that is to be accessed.

Plan

Plan name to be bound to Db2.

Execute Db2 BIND Command

For HFS deployment, indicates whether to bind the plan. Yes is the default value.

DSNCLST Library Name

Name of the Db2 installation DSNCLST library.

This attribute is available only if you choose to bind your program.

ENCODING

Specifies the application encoding for all host variables in the plan. Valid values are ASCII, EBCDIC, UNICODE, ccsid.

DSNLOAD Library Name

Name of the Db2 installation DSNLOAD Library.

If this value was supplied at installation time, it will be displayed here.

Owner

Authorization ID of the Owner of the Plan.

This attribute is available only if you choose to bind your program

Isolation Level

Isolation level. CS is the default value.

This attribute is available only if you choose to bind your program.

Grant

Grants plan execution to public.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down list and enter a name in the Profile Name input box (the extension is added automatically).

Reference: Connection Attributes for Db2 With JDBC

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the JDBC data source.

Driver name

Name of the JDBC driver.

See the driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names that will be available for Java Services. The value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Reference: Connection Attributes for Db2 With SQL

For a full IBM i installation, once the adapter is configured for IBM i you must go to the Change Settings pane and set an isolation level. Failure to set an explicit isolation level causes Db2 to generate multiple commit/rollback warning messages in the adapter's job log. Any isolation level may be selected for the purpose of stopping the multiple warning messages. For more information, see *Controlling Types of Locks on IBM i* on page 534.

Note: In the IBM i environment, isolation level is preset to NC (No Commit) so no action is required.

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Db2, at connection time, for authentication.

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES connection DSN_name/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Db2, at connection time, for authentication.

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES connection DSN_name/
```

Trusted authentication. The adapter connects to Db2 as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

The available parameters are:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES connection DSN_name/,
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

DSN_*name*

Is the Db2 data source name (DSN) you wish to access. It must match an entry in the odbc.ini file.

userid

Is the primary authorization ID by which you are known to Db2.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allows the application to access the Db2 database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES CON01 SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Db2 database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES CON01 SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Db2 database server using operating system authentication:

```
ENGINE DB2 SET CONNECTION_ATTRIBUTES CON01 SAMPLESERVER/,
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Migrate option on the Web Console migrates your server settings to the newer release. To access this option, choose *Workspace*, then *Migrate* from the menu bar. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

  ```
  ENGINE DB2 SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

  ```
  ENGINE DB2 SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
  ```

DB2 CURRENT SQLID (z/OS)

Db2 accepts two types of IDs, the primary authorization ID and one or more optional secondary authorization IDs. It also recognizes the CURRENT SQLID setting.

Any interactive user or batch program that accesses a Db2 subsystem is identified by a primary authorization ID. A security system, such as RACF, normally manages the ID. During the process of connecting to Db2, the primary authorization ID may be associated with one or more secondary authorization IDs (usually RACF groups). Each site controls whether it uses secondary authorization IDs. For more information about using the adapter in conjunction with external security packages, see the server manual for your platform.

The primary authorization ID is the same ID passed to the server at connect time. This user ID is then used to connect to the Db2 subsystem.

The Db2 Data Source Administrator may grant privileges to a secondary authorization ID that are not granted to the primary ID. Thus, secondary authorization IDs provide the means for granting the same privileges to a group of users. (The DBA associates individual primary IDs with a secondary ID and grants the privileges to the secondary ID.)

The DB2 CURRENT SQLID may be the primary authorization ID or any associated secondary authorization ID. At the beginning of the session, the CURRENT SQLID is the primary authorization ID.

Syntax: How to Reset CURRENT SQLID

You can reset the CURRENT SQLID in a stored procedure or a profile using the following adapter command using the following parameters:

```
ENGINE DB2 SET CURRENT SQLID = 'sqlid'
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`sqlid`

Is the primary or secondary authorization ID, which must be enclosed in single quotation marks as shown. All Db2 security rules are respected.

The CURRENT SQLID is the default owner ID for Db2 objects, such as tables or indexes, created with dynamic SQL commands. The CURRENT SQLID is also the sole authorization ID for GRANT and REVOKE commands. It must have all the privileges needed to create objects as well as GRANT and REVOKE privileges. In addition, the CURRENT SQLID is the implicit owner for unqualified table names.

Other types of requests, such as SQL SELECT, INSERT, UPDATE, or DELETE requests, automatically search for the necessary authorization using the combined privileges of the primary authorization ID and all of its associated secondary authorization IDs, regardless of the DB2 CURRENT SQLID setting.

The CURRENT SQLID setting remains in effect until the thread to Db2 is disconnected, when it reverts to the primary authorization ID.

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE DB2 SET DEFAULT_CONNECTION connection
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: **Selecting the Default Connection**

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE DB2 SET DEFAULT_CONNECTION SAMPLE
```

Controlling Connection Scope

The SET AUTODISCONNECT and AUTOCLOSE commands control the persistence of connections when using the adapter.

Controlling Connection Scope With AUTOCLOSE (z/OS)

SET AUTOCLOSE initiates the Db2 Call Attachment Facility (CAF) CLOSE operation. It determines how long a thread (the connection between the application program in the user's address space and the Db2 application plan) is open. The thread is not the same as the address space connection to Db2; that connection is controlled by the AUTODISCONNECT setting.

In the server, an application program is one of the following:

- The dynamic Adapter for Db2.

- A CALLPGM subroutine using embedded SQL (non-CLI).

Generally speaking, each program has a corresponding plan or package.

A site that installs a Db2 subsystem determines the maximum number of concurrent users (threads) the subsystem will support. Since each user requires enough virtual storage for their application plan, this setting controls the amount of storage the site wants to allocate to active Db2 users at any one time.

The CAF CLOSE command deallocates the Db2 thread, releasing the virtual storage for the application plan. Db2 requires that an existing thread to a plan be closed before a thread to another plan is opened. If a thread is closed without a subsequent OPEN operation, the closed thread becomes "inactive"; the user is still connected to Db2, but not to a particular application plan. The user (task) still owns the thread; it is not available to other users. To release the thread, the user must disconnect completely from Db2.

Note: The term pseudo-conversational describes the type of transaction processing provided when you use AUTOCLOSE ON COMMIT.

Syntax: How to Control Connection Scope With AUTOCLOSE

```
ENGINE DB2 SET AUTOCLOSE ON {FIN|COMMIT}
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`AUTOCLOSE`

Issues the Db2 Call Attach Facility (CAF) CLOSE operation.

`FIN`

Disconnects automatically only after the server session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Controlling Connection Scope With AUTODISCONNECT

AUTODISCONNECT completely detaches the users address space (or task) from Db2. After a DISCONNECT, the task must re-establish its connection to Db2 before performing any data source work. The tasks that frequently issue the DISCONNECT command are connected to Db2 for shorter periods of time, allowing other tasks to connect and acquire threads as needed. However, there is significant system overhead associated with frequently connecting and disconnecting, and the possibility exists that no thread will be immediately available when the task attempts to reconnect.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Control Connection Scope With AUTODISCONNECT

```
ENGINE DB2 SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Db2 Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Db2 data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Db2 table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 525.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Applications or Adapters pages of the Web Console.

1. From the Web Console Applications page, click *Get Data*.
2. Right-click a connection for a configured adapter.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click the *Add* button.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym is created and added under the specified application directory.

Note:

- When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Db2

The following list describes the synonym creation parameters for which you can supply values.

Object Type

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

- If you select *Aliases* as a filtering option and the Alias points to a Remote Db2 system, it must be Version 8 or higher. Only one level of Remote Db2 is supported.
- If you select *Nicknames* as a filtering option on platforms where Db2 supports the Nickname object type (Windows and UNIX), a basic synonym is created for this type of Db2 object; no Titles, Remarks, or Keys are recorded in the synonym. The user ID who creates the synonym must have SELECT authority for this object type.

Important: If you select Stored Procedures as your object type, the input parameters will be a little different from those described here. For details, see *Reporting Against a Db2 Stored Procedure* on page 525.

Owner/Schema

Selecting this option adds the Owner/Schema and Object Name parameters to the screen. Select an owner/schema from the drop-down list or type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. Then click *Search*. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

Object Name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen. Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. Then click *Search*. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Filter by Library/Object Name (IBM i)

To avoid the return of an extremely large and potentially unmanageable list, always supply a value for Library or Object Name:

- **Library.** Type a string for filtering the Library (or Db2 Collection), inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the table, view, or object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables, views, or objects whose names begin with the letters ABC; %ABC to select all tables, views, or objects whose names end with the letters ABC; %ABC% to select all tables, views, or objects whose names contain the letters ABC at the beginning, middle, or end.

Note: When you create a synonym for Db2 on the IBM i platform, standard IBM i naming conventions apply to the target data source. Therefore, the Adapter for Db2 supports the use of double-quotation marks around any library name and/or file name that contains lowercase or NLS characters.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Row Limit

Select the number of objects to display on the Create Synonym page.

For Subquery

Only available when *External SQL Scripts* is selected from the object type drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 1211.

Create: Cluster Synonym or Base Synonyms

Select the button for the type of synonym you want to create.

One-part name (IBM i)

On the IBM i platform, the One-part name check box is unchecked by default. The unchecked behavior generates a table name that includes the explicit name of the library containing the table. For example, if you specified a library on the first Create Synonym pane, a qualified name like the following is automatically created in the Access File for a TABLE or VIEW:

```
TABLENAME=MYLIB/MYTABLE
```

For a Stored Procedure, this would be a Db2 RPC and identified in the Access File as:

```
STPNAME=MYLIB/MYPROC
```

With this explicit type of entry in the Access File, at run-time the library is directly referenced and the object opened.

If you select the check box, the explicit library name is not stored in the metadata (Access File). When the synonym is generated, the library portion of the table name is omitted from the Access File, and appears as follows:

```
TABLENAME=MYTABLE
```

or

```
STPNAME=MYPROC
```

With this type of entry in the Access File (one-part name), the Db2 library search path will be used at run time. The exact composition varies depending on if Db2 is configured as a CLI or SQL connection.

For CLI configured servers, the search path will be the default library of Db2 for a CLI connected user (usually QSYS and QSYS2 plus the default library of the user). It is also controllable by use of a passthru command setting the path, such as SLQ DB2 SET PATH “QSYS”,”QSYS2”,”FOO”,”EXTRAFOO” as issued within a requests FOCEXCEC procedure or a profile. For SQL configured servers, the search path will be *LIBL. The library path of the process can be controlled by commands, such as ADDLIBLE, either before server start or as issued within a requests FOCEXCEC procedure or a profile.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Fact or Dimension

You can check Fact or Dimension to generate the segment as a fact table or a dimension segment. If you are creating a cluster synonym, you can right-click a selected fact table and select *Show Related Dimensions* or *Add Related Dimensions* to show a list of related dimensions or add related dimensions to the synonym.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- When creating base synonyms, you can select all tables in the list by clicking the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Once you have selected the objects for which you want to create synonyms, click the *Create Base Synonyms* or *Create Cluster Synonym* button on the ribbon.

Example: Sample Generated Synonym

An Adapter for Db2 synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=DB2 ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4  ,MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25 ,MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4  ,MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,WRITE=YES,$
```

Reference: Mapping Db2 Table and Column Attributes Into a Synonym

These mappings are OS-system specific:

Mapping Db2 Table Attributes

Platform	Db2 Table Attribute	Mapping in Synonym	Notes
UNIX/Windows	COMMENT	REMARKS	
IBM i	REMARK LABEL	REMARKS	The synonym creation facility picks up a Db2 table TABLE level REMARK or LABEL value, whichever is not null, for use as a Master File REMARKS= value. If the Db2 table has both a populated REMARK and a populated LABEL, the REMARK value will be used.
z/OS	COMMENT LABEL	REMARKS	The synonym creation facility picks up a Db2 table TABLE level COMMENT or LABEL value, whichever is not blank, for use as a Master File REMARKS= value. If the Db2 table has both a populated COMMENT and a populated LABEL, the COMMENT value will be used.

Mapping Db2 Column Attributes

Platform	Db2 Column Attribute	Mapping in Synonym	Notes
UNIX/Windows	COMMENT	DESCRIPTION	
IBM i	LABEL COMMENT	TITLE DESCRIPTION	

Platform	Db2 Column Attribute	Mapping in Synonym	Notes
z/OS	LABEL COMMENT	TITLE DESCRIPTION	

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: TABLENAME=[*location.*][*owner.*]*tablename* **Note:** Location is valid only with Db2 CAF and specifies the subsystem location name. For IBM i, the syntax is: TABLENAME=[*library/*]*tablename*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
DBSPACE	Optional keyword that indicates the storage area for the table. For example: *datasource.tablespace* DATABASE *datasource*

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	Set to Yes to enable autoincrementing.

Keyword	Description
START	Initial value in incrementing sequence
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Db2 Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Db2 data types VARCHAR. By default, the server maps this data type as variable character (A*n*V).

The following table lists data type mappings based on the value of VARCHAR.

		VARCHAR ON		VARCHAR OFF	
Db2 Data Type	**Remarks**				
VARCHAR (*n*)	*n* is an integer between 1 and 32768	A*n*V	A*n*V	A*n*	A*n*

Syntax: **How to Control the Mapping of Variable-Length Data Types**

```
ENGINE DB2 SET VARCHAR {ON|OFF}
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Db2 data type VARCHAR as variable-length alphanumeric (A*n*V). This is required for Unicode environments. ON is the default value.

`OFF`

Maps the Db2 data type VARCHAR as alphanumeric (A).

BLOB Activation

Db2 data types that support the *for bit data* attribute including VARCHAR(n), where *n* > 256 and LONG VARCHAR, can be supported in the server as Binary Large Objects (BLOBs). This support is for both read and write access.

Syntax: How to Activate BLOB

To activate this support, you must issue the following command in one of the supported server profiles

```
ENGINE DB2 SET CONVERSION LONGCHAR BLOB
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`BLOB`

Activates long binary support. ALPHA is the default value.

BLOB Read/Write Support

For Db2 data types VARCHAR (>256) and LONG VARCHAR which have the *for bit data* attribute, the server provides read and write support using three server remote procedures routines. These routines are:

Routine	Used to...
EDABS	Send binary image data to the server.
EDABE	Mark the end of the binary image.
EDABK	Purge the binary image from server storage.

The sequence of operations for the client application is:

1. Converts every binary byte of the image into 2 bytes of hexadecimal data and stores the result in an internal buffer. If the image is large, this could be split into manageable pieces.
2. Sends the converted binary bytes to the server using remote procedure EDABS. The server converts the hex data back to binary and stores the image ready for INSERT/UPDATE into Db2.
3. Repeats steps 1 and 2 until the complete image is sent to the server in hex format. It then sends remote procedure EDABE to mark the end of the image.
4. The client application prepares an SQL INSERT/UPDATE using parameter markers for the columns of the row.
5. The client application issues a BIND for the columns using CHAR(16) for the image column.
6. The client application issues an EXECUTE USING command giving the data values for the row columns but using 'BLOB' for the image. The row will be INSERTed/UPDATEd using the image buffer stored on the server.
7. Once the application has finished with the stored image on the server (and COMMITed the data), it should send the EDABK Remote Procedure to release server storage.

For full details and examples of how to maintain Db2 *for bit data* columns, see the *API Reference* and *Connector for ODBC* manuals.

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE DB2 SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96

Reporting Against a Db2 Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Db2 stored procedures and report against the output parameters and answer set of a procedure. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters, OUT parameters, and INOUT parameters in OUT mode, as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 525.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 529.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

A synonym describes the stored procedure parameters and answer set.

An answer set structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, "input parameters" refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception: if you know the internal logic of the procedure, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

In IBM i, the metadata of a stored procedure that has an answer set will include column descriptions. These will appear as title fields in the synonym, as shown in the following example:

```
FILENAME=TEST_DESCRIPTIONS, SUFFIX=DB2 , $
  SEGMENT=INPUT, SEGTYPE=S0, $
    FIELDNAME=, ALIAS=DUMMY, USAGE=A1, ACTUAL=A1, MISSING=ON, $
  SEGMENT=ANSWERSET1, SEGTYPE=S0, PARENT=INPUT, $
    FIELDNAME=STORECODE, ALIAS=STORECODE, USAGE=A6, ACTUAL=A6,
      TITLE='Store,Code', $
    FIELDNAME=STORENAME, ALIAS=STORENAME, USAGE=A30, ACTUAL=A30,
      MISSING=ON, TITLE='Store,Name', $
    FIELDNAME=COUNTRY, ALIAS=COUNTRY, USAGE=A15, ACTUAL=A15,
      MISSING=ON, TITLE='Country', $
    FIELDNAME=REGION, ALIAS=REGION, USAGE=A25, ACTUAL=A25,
      MISSING=ON, TITLE='Region', $
```

Example: Synonym for Db2 Stored Procedure CustOrders

The following synonym describes a stored procedure, DB2SPR04 SP, with IN/OUT OUTPUT parameters.

```
FILENAME=SDB2SPR04, SUFFIX=DB2     , $
  SEGMENT=INPUT, SEGTYPE=S0, $
    FIELDNAME=PARM1, ALIAS=P0001, USAGE=I11, ACTUAL=I4,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM2, ALIAS=P0002, USAGE=I6, ACTUAL=I4,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM5, ALIAS=P0005, USAGE=YYMD, ACTUAL=DATE,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM6, ALIAS=P0006, USAGE=HHIS, ACTUAL=HHIS,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM7, ALIAS=P0007, USAGE=HYYMDm, ACTUAL=HYYMDm,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM11, ALIAS=P0011, USAGE=P17.5, ACTUAL=P8,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM13, ALIAS=P0013, USAGE=P20, ACTUAL=P10,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM15, ALIAS=P0015, USAGE=F9.2, ACTUAL=F4,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PARM16, ALIAS=P0016, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $


  SEGMENT=OUTPUT, SEGTYPE=S0, PARENT=INPUT, $
    FIELDNAME=RETURN_CODE, ALIAS=P0000, USAGE=I11, ACTUAL=I4,
      MISSING=ON, TITLE='Return Code', $
    FIELDNAME=PARM1, ALIAS=P0001, USAGE=I11, ACTUAL=I4, MISSING=ON, $
    FIELDNAME=PARM2, ALIAS=P0002, USAGE=I6, ACTUAL=I4, MISSING=ON, $
    FIELDNAME=PARM3, ALIAS=P0003, USAGE=I11, ACTUAL=I4, MISSING=ON, $
    FIELDNAME=PARM4, ALIAS=P0004, USAGE=I6, ACTUAL=I4, MISSING=ON, $
    FIELDNAME=PARM5, ALIAS=P0005, USAGE=YYMD, ACTUAL=DATE, MISSING=ON, $
    FIELDNAME=PARM6, ALIAS=P0006, USAGE=HHIS, ACTUAL=HHIS,
      MISSING=ON, $
    FIELDNAME=PARM7, ALIAS=P0007, USAGE=HYYMDm, ACTUAL=HYYMDm,
      MISSING=ON, $
    FIELDNAME=PARM8, ALIAS=P0008, USAGE=YYMD, ACTUAL=DATE, MISSING=ON, $
    FIELDNAME=PARM9, ALIAS=P0009, USAGE=HHIS, ACTUAL=HHIS, MISSING=ON, $
    FIELDNAME=PARM10, ALIAS=P0010, USAGE=HYYMDm, ACTUAL=HYYMDm,
      MISSING=ON, $
    FIELDNAME=PARM11, ALIAS=P0011, USAGE=P17.5, ACTUAL=P8, MISSING=ON, $
    FIELDNAME=PARM12, ALIAS=P0012, USAGE=P17.5, ACTUAL=P8, MISSING=ON, $
    FIELDNAME=PARM13, ALIAS=P0013, USAGE=P20, ACTUAL=P10, MISSING=ON, $
    FIELDNAME=PARM14, ALIAS=P0014, USAGE=P20, ACTUAL=P10, MISSING=ON, $
    FIELDNAME=PARM15, ALIAS=P0015, USAGE=F9.2, ACTUAL=F4, MISSING=ON, $
    FIELDNAME=PARM16, ALIAS=P0016, USAGE=D20.2, ACTUAL=D8, MISSING=ON, $
    FIELDNAME=PARM17, ALIAS=P0017, USAGE=F9.2, ACTUAL=F4, MISSING=ON, $
    FIELDNAME=PARM18, ALIAS=P0018, USAGE=D20.2, ACTUAL=D8, MISSING=ON, $
```

Reference: Synonym Creation Parameters for Stored Procedures

The following list describes the synonym creation parameters for a stored procedure.

Object Type

Select *Stored Procedures*.

Owner/Schema and Object name (all platforms except IBM i)

Selecting these options adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the procedure names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all procedures whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Library

Object Name (IBM i)

To avoid the return of an extremely large and potentially unmanageable list, always supply a value for Library or Object Name:

- **Library.** Type a string for filtering the Library (or Db2 Collection), inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the table, view, or object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables, views, or objects whose names begin with the letters ABC; %ABC to select all tables, views, or objects whose names end with the letters ABC; %ABC% to select all tables, views, or objects whose names contain the letters ABC at the beginning, middle, or end.

Select

Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.

Synonym Name

The name of the synonym, which defaults to the stored procedure name.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters.

If all procedures have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Creating a Report Against a Stored Procedure

You can report against a stored procedure's answer set using the same facilities you use to report against a database table:

- **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 530.
- **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 529.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

IF

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters, or that there are no required parameters.

`WHERE`

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode. You must specify the value of each parameter on a separate line.

`in-parameter`

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

`value`

Is the value you are passing to a parameter.

Customizing the Db2 Environment

The Adapter for Db2 provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Improving Response Time

The Adapter for Db2 on z/OS supports parallel processing if you issue the SET CURRENT DEGREE command prior to the request.

Syntax: **How to Improve Response Time**

```
ENGINE DB2 SET CURRENT DEGREE {'1'|'ANY'}
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`1`

Invokes serial processing. 1 is the default value.

ANY

Invokes parallel processing for dynamic requests. If the thread to Db2 is closed during the session, the value resets to 1.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create. For the duration of the session, the adapter places these tables in the Db2 tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, Db2 uses the default tablespace for the connected user.

Syntax: How to Designate a Default Storage Space for Tables

```
ENGINE DB2 SET DBSPACE {datasource.tablespace|DATABASE datasource}
```

where:

DB2

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

datasource

Is the data source name. DSNDB04 is the default value, which is a public data source.

tablespace

Is a valid table space name in the data source.

Note: This command will only affect CREATE FILE requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

On Linux, UNIX, and Windows, if you want to place a volatile or global temporary table in a user-created tablespace, issue the following command.

```
ENGINE DB2 SET DBSPACE usertemp
```

where:

usertemp

Is the name of a user temporary tablespace.

Note: When this setting is in effect, all tables created in the session will automatically be created in this tablespace. However, permanent tables cannot be placed in this type of tablespace and will result in an error. Therefore, you should not place this setting in any profile. You can add it to a procedure creates a temporary table. After the procedure executes, the setting will no longer be in effect.

Controlling the Types of Locks

You can use the SET ISOLATION command to specify the isolation level of transactions created by the adapter. The isolation level controls the types of locks for objects referenced in the requests executed within the transaction.

If you are working in the IBM i environment, see *Controlling Types of Locks on IBM i* on page 534 for OS-specific variations of SET ISOLATION syntax. For related information about another use for the SET ISOLATION syntax, see *Creating and Updating Db2 Files on IBM i* on page 535.

Syntax: How to Control the Lock Type

```
ENGINE DB2 SET ISOLATION level
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`level`

Sets the Db2 isolation level, which is mapped to the IBM RDBMS isolation level. If you do not specify an isolation level, the level is reset to the adapter default.

Db2 Isolation Level (CLI)	IBM RDBMS Isolation Level
RC (SQL_TXN_READ_COMMITTED)	CS (Cursor Stability)
SE (SQL_TXN_SERIALIZABLE_READ)	RR (Repeatable Read)
RR (SQL_TXN_REPEATABLE_READ)	RS (Read Stability)
RU (SQL_TXN_READ_UNCOMMITTED)	UR (Uncommitted Read)

Db2 Isolation Level (CLI)	IBM RDBMS Isolation Level
NC (SQL_TXN_NO_COMMIT)	Not applicable

- **RC.** Releases shared locks as the cursor moves on in the table. Use for read-only requests. RC is the default value. Maps to CS (Cursor Stability).
- **SE.** Locks the retrieved data until it is released by an SQL COMMIT WORK or SQL ROLLBACK WORK statement. Maps to RR (Repeatable Read).
- **RR.** Maps to RS (Read Stability). For more information, see the *Db2 Command and Utility Reference*.
- **RU.** Provides read-only access to records even if they are locked. However, these records may not yet be committed to the data source. Maps to UR (Uncommitted Read).
- **NC.** Commit and rollback operations have no effect on SQL statements. Any changes are effectively committed at the end of each successful change operation and can be immediately accessed. For more information, see *Creating and Updating Db2 Files on IBM i* on page 535 and the *Db2 Command and Utility Reference*.

Note:

- The adapter does not validate the isolation level values. If you issue the SET ISOLATION command with an invalid value, the adapter will not return a message.
- To display the isolation level setting, issue the ENGINE DB2 ? CONNECTINFO query command.

Reference: Controlling Types of Locks on IBM i

For the IBM i environment, the isolation level is preset to NC (no commit) so no action is required.

For a full IBM i installation, two forms of the SET ISOLATION command are supported: a long form that executes immediately in passthru mode, and a short form that is held until an actual SQL request is processed.

The two command variations are very similar, however, the short form, under an SQL adapter configuration (rather than a CLI configuration), generates the following Db2 message in the IBM i system process log if a COMMIT, ROLLBACK, or SAVEPOINT command is issued before any real work is done (possibly due to an AUTOCOMMIT command):

```
SQL7007: COMMIT, ROLLBACK, or SAVEPOINT not valid
```

The syntax variations are:

Short Form

```
ENGINE DB2 SET ISOLATION level
```

Long Form

```
ENGINE DB2 SET TRANSACTION ISOLATION LEVEL level
```

Standard options for *level* apply (see *How to Control the Lock Type* on page 533). However, the instruction to display the current isolation level by issuing the following query command applies only for the *short form* setting:

```
ENGINE DB2 ? CONNECTINFO
```

There is no way to display the current long form setting.

Reference: Creating and Updating Db2 Files on IBM i

The following information applies to HOLD FORMAT DB2 and procedures that update a Db2 object in a non-journaled collection.

While Db2 on IBM i supports CREATE TABLE operations to a non-journaled collection (a library with no journal receivers), Db2 normally considers this a commitment control error and issues an error message. When a HOLD FORMAT DB2 command is issued, the same error condition triggers an error message to the adapter. In response, the adapter creates the table, but does not perform the load step. However, if the server is configured with Db2 as a CLI-based adapter, you can use the ISOLATION setting of NC (No Commit) to prevent Db2 from triggering the error message, thereby enabling the table to load.

You can set ISOLATION to NC on a request-by-request basis before issuing HOLD FORMAT DB2:

```
SQL DB2 SET ISOLATION NC
```

Alternatively, you can set the NC option server wide from the Adapter for Db2's Change Settings pane. (To access this pane, click *Adapters* on the menu bar, right-click the name of the configured adapter, and choose *Change Settings* from the menu.)

After completing this task, revert to the original ISOLATION setting, if appropriate.

Note: An error message like the following will be generated if you issue a HOLD FORMAT DB2 ... DROP command but do not have the authority to DROP an existing file.

```
(FOC1400) SQLCODE IS -601 (HEX: FFFFFDA7): [42710] ABC in XYZ type *FILE
already exists.
```

```
(FOC1414) EXECUTE IMMEDIATE ERROR.
```

- If you receive this message, make sure that you have DB2 DROP/CREATE authority and reissue the command.
- When connecting from a non-IBM i platform through a local Db2 client, the SET isolation does not work. You can use the following specific platform syntax.

Note:

- **From a Windows platform:**

  Add a CLI parameter *TxnIsolation* with the value *32* within your ODBC settings under *Administrative Tools*.

- **From a Unix/Linux platform:**

  You can run the following Db2 command on your database:

  ```
  DB2 update CLI CFG for section <tablename> using TxnIsolation 32
  ```

  Verify these settings with the following command:

  ```
  DB2 get CLI CFG
  ```

  As an SQL alternative (not OS-specific), you can add `WITH NONE` to the end of your SQL UPDATE command.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the index space implicitly created by the CREATE FILE and HOLD FORMAT commands.

Syntax: How to Set IXSPACE

```
ENGINE DB2 SET IXSPACE [index-spec]
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

index-spec

Is the portion of the DB2 CREATE INDEX statement that defines the parameters for the index. It can consist of up to 94 bytes of valid Oracle index space parameters. To reset the index space parameters to their default values, issue the SET IXSPACE command with no parameters.

Note: Refer to the Db2 documentation for more information on this command.

The long form of SQL Passthru syntax for commands exceeding one line is:

```
ENGINE DB2
SET IXSPACE index-spec
END
```

For example, to specify the NOSORT, NOLOGGING, and TABLESPACE portions of the DB2 CREATE INDEX statement, enter the following commands:

```
ENGINE DB2
SET IXSPACE NOSORT NOLOGGING
TABLESPACE TEMP
END
```

Note: This command will only affect CREATE INDEX requests issued by CREATE FILE and HOLD FORMAT DB2 commands. It does not affect Passthru CREATE INDEX commands, for example:

```
ENGINE DB2 SET IXSPACE TABLESPACE tablespace_name
TABLE FILE table_name
PRINT *
ON TABLE HOLD AS file_name FORMAT DB2
END
```

Activating NONBLOCK Mode

The Adapter for Db2 has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Setting*s from the menu. The Change Settings pane opens.

Note: This setting is not supported for the z/OS Unified Server Db2 CAF Adapter (all deployment modes).

Syntax: How to Activate NONBLOCK Mode

```
ENGINE DB2 SET NONBLOCK {0|n}
```

where:

DB2

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Controlling Column Names

You can use the SET NOCOLUMNTITLE command to control the column names in a report when executing a stored procedure.

Syntax: How to Control Column Names

```
ENGINE DB2 SET NOCOLUMNTITLE {ON|OFF}
```

where:

DB2

Indicates the adapter. You can omit this parameter value if you previously issued the SET SQLENGINE command.

ON

Uses generated column names (for example, E01, E02, and so on) instead of the column names returned by Db2.

OFF

Uses the column names returned by Db2. OFF is the default value.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE DB2 SET PASSRECS {ON|OFF}
```

where:

DB2

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Note: This behavior applies for all supported versions of Db2 up to Version 9 on all platforms. In Db2 Version 9 on z/OS, the PASSRECS command does not provide a row count for mass DELETE operations (that is, DELETE statements without a WHERE clause). However, you can obtain a row count in this situation by adding a WHERE phrase to your report request. For example, the following request would generate a delete count:

```
SQL DB2
DELETE FROM DB2TAB WHERE F1=15;
END
FOC1364) ROWS AFFECTED BY PASSTHRU COMMAND: 1/DELETE
```

Setting End-User Information

In the UNIX and Windows Reporting Server environment, you can set values for end-user information to be passed to a Db2 server when the next SQL request is processed. This information includes the:

- Client user ID
- Application program name
- Workstation name
- Accounting string

You can then query the information using SELECT statements.

Note: For a Db2 server on z/OS, you can also set the end-user information but only from a UNIX or Windows server connecting to a Db2 on z/OS.

Syntax: How to Set End-User Information

```
ENGINE DB2 SET CLIENT_APPLNAME application_name
ENGINE DB2 SET CLIENT_USERID userid
ENGINE DB2 SET CLIENT_WRKSTNNAME workstation
ENGINE DB2 SET CLIENT_ACCTNG account
```

where:

application_name

Is the name of an application program.

userid

Is the user ID of a client.

workstation

Is the name associated with the user workstation.

account

Is an accounting string associated with the client user.

Syntax: How to Query End-User Information

In the UNIX and Windows Reporting Server environment, depending on where the Db2 server is running, the end-user information can be queried using the syntax below.

If the Db2 server is on UNIX or Windows:

```
ENGINE DB2
SELECT CLIENT APPLNAME, CLIENT USERID, CLIENT WRKSTNNAME, CLIENT ACCTNG
FROM SYSIBM.SYSDUMMY1;
END
```

If the Db2 server is on z/OS:

```
ENGINE
DB2  SELECT CURRENT CLIENT_APPLNAME,CURRENT CLIENT_USERID,CURRENT
CLIENT_WRKSTNNAME, CURRENT CLIENT_ACCTNG  FROM SYSIBM.SYSDUMMY1;
END
```

Note: Natively on z/OS, you can also view end-user information by using spufi to submit the following command:

```
-DISPLAY THREAD(*) DETAIL
```

Setting Naming Conventions

Controls the separator character used for interpreting multipart table, view, and RPC names. This option is typically used to allow a request coded with "." as the separator to be run on systems that use "/" as the separator (for example, IBM i).

This setting is only valid for use with a server configured as CLI. (This applies to most platforms, including IBM i if so configured). If the server needs to support both naming conventions, you can configure an additional service block with a service block profile that uses the desired naming setting and sends requests to the desired block.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Set Naming Conventions

The available parameters are:

```
ENGINE DB2 SET NAMING {SQL|SYS}
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`SQL`

Standard "." character is used as separator in multipart table names.

SYS

System "/" character is used as separator in multipart table names.

Controlling HOLD DBMS Creation

An extension of the HOLD AS *app/name* FORMAT DB2 syntax enables you to exercise more precise control over the creation of HOLD DBMS files.

Syntax: How to Control DBMS Creation

```
HOLD AS app/name FORMAT DB2 [TABLENAME dbms_name][CONNECTION conn_name]
  [DROP]
```

where:

dbms_name

Is the DBMS table to create. It may be a one, two, or three part name, using the separator appropriate to the DBMS (typically "." (dot)).

For IBM i, a "/" (slash) is the separator, unless the SET NAMING SQL command is being used to reset the separator character.

Note: For Db2 running on IBM i, if the TABLENAME parameter and value are omitted, the default location for the resulting table is the user's default login library. This applies to adapters configured for SQL and CLI.

conn_name

Is the DBMS connection name. When multiple DBMS connections have been configured and are in use, conn_name specifies which connection to use.

DROP

Drops the table before creation. This option enables you to delete either a known table or one that was created and stored in a temporary space when persistValue=*global_temporary*. Without the DROP option, if a table already exists, an error occurs when the table is created, resulting in failure to load the table.

If no table exists, this option is ignored.

Db2 Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Optimizing Non-Equality WHERE-Based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

`LEFT_OUTER`

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

`hostfile`

Is the host Master File.

`AT`

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

`hfld1`

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

`tag1`

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

`WITH hfld2`

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

`MULTIPLE`

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

`UNIQUE`

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations; it always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: Conditions for WHERE-Based Outer Join Optimization

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left-Outer Join

The following request creates a left outer conditional join between two Db2 data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE baseapp/EQUIP AT CARS
TO ALL FILE baseapp/CARREC AT CARC
WHERE CARS NE CARC;
END
TABLE FILE baseapp/EQUIP
PRINT CARS CARC STANDARD
BY MODEL
END
```

The WebFOCUS request is translated to a single DB2 SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."CARS"(CHAR( 16)),T1."STANDARD"(CHAR( 40)),
T2."CARC"(CHAR( 16)),T2."MODEL"(CHAR( 24)) FROM ( EQUIP T1 LEFT
OUTER JOIN CARREC T2 ON (T1."CARS" <> T2."CARC") ) ORDER BY
T2."MODEL";
```

Specifying the Block Size for Retrieval Processing

The Adapter for Db2 supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE DB2 SET FETCHSIZE n
```

where:

`DB2`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques for the CLI adapter or a cursor with Rowset positioning and multi row Fetch for the CAF adapter. Accepted values are 1 to 32000. The default is 100. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE DB2 SET INSERTSIZE n
```

where:

DB2

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE DB2 SET FASTLOAD [ON|OFF]
```

where:

DB2

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Uses the Bulk Insert API. ON is the default.

OFF

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Db2 (Windows and UNIX).

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Improving Efficiency With Aggregate Awareness

Aggregate awareness substantially improves the efficiency of queries.

For details about this feature, see *Aggregate Awareness Support* on page 2693.

Syntax: How to Set Aggregate Awareness

```
SET AGGREGATE_AWARENESS {FRESH_ONLY|OLD_OK|OFF}
```

where:

FRESH_ONLY

Sets different values for the parameters associated with each RDBMS.

OLD_OK

Sets different values for the parameters associated with each RDBMS.

OFF

If no option is selected, the behavior of the target RDBMS is determined by the database configuration options. There is no default for this setting.

For details about adapter-specific settings, see *Usage Notes for Aggregate Awareness* on page 2694.

Using Db2 Cube Views

The Adapter for Db2 supports Cube Views as an object type. You can specify Cubes as the object type for which you want to create synonyms.

Before you can use the Adapter for Db2 with Cube Views, you must define Db2 connections for the database(s) containing the Db2 cube views.

For details, see *Creating Synonyms* on page 512.

Mapping Metadata for Db2 Cubes Views

Synonyms are generated for Cubes (not Cube Models). These synonyms reflect the structure of the cube and its attributes and measures, rather than the structure of the underlying Db2 tables and their columns.

The Master File synonym component has several segments:

- The root segment corresponds to the Cube Facts.
- The children segments correspond to the Cube Hierarchies.

The fields in the root segment reflect the Cube Measures. Field names and titles are generated based on measure business names, while aliases are generated based on measure names.

- The PROPERTY attribute for these fields contains the word Measure (or Calculated Measure for calculated measures).
- The REFERENCE attribute contains the aggregated expression for measures that are based on simple columns (not expressions) and have aggregations that match those available in the DML.

Example: Sample Db2 Cube View Master File

```
FILENAME=BASEAPP/COMPLEX_MEASURES__C_, SUFFIX=DB2CV   , $
  SEGMENT=SALESCF, SEGTYPE=S0, $
    FIELDNAME=4ADDS, ALIAS=4adds, USAGE=I11, ACTUAL=I4, MISSING=ON,
      TITLE='4adds', REFERENCE=CNT.4ADDS, PROPERTY=MEASURE,  $
    FIELDNAME=POWEROFADDS, ALIAS=powerofadds, USAGE=D21.2, ACTUAL=D8,
      MISSING=ON, TITLE='powerofadds', PROPERTY=MEASURE,  $
    FIELDNAME=PROFIT_CONST, ALIAS=profit*const, USAGE=D21.2, ACTUAL=D8,
      MISSING=ON, TITLE='profit*const', PROPERTY=MEASURE,  $
    FIELDNAME=RANDOM, ALIAS=random, USAGE=I11, ACTUAL=I4, MISSING=ON,
      TITLE='random', REFERENCE=CNT.RANDOM, PROPERTY=MEASURE,  $
    FIELDNAME=3_COLS_WITH_MIXTUREOF_FUNC,
      ALIAS='3 cols with mixtureof FUNC', USAGE=D21.2, ACTUAL=D8,
      MISSING=ON, TITLE='3 cols with mixtureof FUNC',
      PROPERTY=MEASURE,  $
    FIELDNAME=ROUND_OF_ANOTHER_MEAS, ALIAS='round of another meas',
      USAGE=D21.2, ACTUAL=D8, MISSING=ON, TITLE='round of another meas',
      PROPERTY=MEASURE,  $

  SEGMENT=PRODUCT__CH_, SEGTYPE=U, PARENT=SALESCF, $
    FIELDNAME=PRODUCT_GROUP_ID, ALIAS=product_group_ID, USAGE=I11,
      ACTUAL=I4, MISSING=ON, TITLE='product_group_ID',
      WITHIN='*product (CH)', $
    FIELDNAME=PRODUCT_LINE_ID, ALIAS=product_line_ID, USAGE=I11, ACTUAL=I4,
      MISSING=ON, TITLE='product_line_ID', WITHIN=PRODUCT_GROUP_ID, $
    FIELDNAME=PRODUCT_LINE_NAME,
      ALIAS='12  product_line_ID   product_line_name', USAGE=A25V,
      ACTUAL=A25V, MISSING=ON, TITLE='product_line_name',
      REFERENCE=PRODUCT_LINE_ID, PROPERTY=CAPTION,  $

  SEGMENT=STORE__CH_, SEGTYPE=U, PARENT=SALESCF, $
    FIELDNAME=STORE_LOCATION_ID, ALIAS='**** store_location_ID_1',
      USAGE=I11, ACTUAL=I4, MISSING=ON, TITLE='store_location_ID',
      WITHIN='*store (CH)', $
    FIELDNAME=STORE_ADDRESS,
      ALIAS='store_location_ID_1   _store_address', USAGE=A25V,
      ACTUAL=A25V, MISSING=ON, TITLE='store address',
      REFERENCE=STORE_LOCATION_ID, PROPERTY=UDA,  $
    FIELDNAME=STORE_CITY, ALIAS='store_location_ID_1   _store_city',
      USAGE=A45, ACTUAL=A45, MISSING=ON, TITLE='store_city',
      REFERENCE=STORE_LOCATION_ID, PROPERTY=UDA,  $
    FIELDNAME=STORE_ID, ALIAS=store_ID, USAGE=I11, ACTUAL=I4, MISSING=ON,
      TITLE='store_ID',WITHIN=STORE_LOCATION_ID, $
```

Calling a Db2 Stored Procedure Using SQL Passthru

Db2 stored procedures are supported using SQL Passthru. These procedures need to be developed within Db2 using the CREATE PROCEDURE command.

The adapter supports stored procedures with IN, OUT, and INOUT parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets are returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to the rules of the underlying DBMS. Note that the examples shown in this section are SQL-based. See the DBMS documentation for rules, languages, and additional programming examples.

Syntax: How to Invoke a Stored Procedure

```
SQL DB2 EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

`DB2`

Is the ENGINE suffix for Db2.

`procname`

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

You can employ either SQL or SYS naming conventions to control the separator character used for interpreting multipart names, as described in *Setting Naming Conventions* on page 541.

`parameter_specification`

IN, OUT, and INOUT parameters are supported. Use the variation required by the stored procedure:

`IN`

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

OUT

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

INOUT

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

Note that this sample employs the SQL naming convention, where the "." character is used as the separator in multipart table names. If your site uses the SYS[TEM] naming convention (typical in IBM i environments), the "/" character is used as a separator in multipart table names. In this case, adjust the separator character as needed to conform to the SYS naming convention. For details, see *Setting Naming Conventions* on page 541.

In this example, a user invokes a stored procedure, edaqa.test_proc01, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL DB2 EX edaqa.test_proc01 125,?,?/3.14,,'abc',,'xyz'
END
```

Example: Sample Stored Procedure

Note that this sample employs the SQL naming convention, where the "." character is used as the separator in multipart table names. If your site uses the SYS[TEM] naming convention (typical in IBM i environments), the "/" character is used as a separator in multipart table names. In this case, adjust the separator character as needed to conform to the SYS naming convention. For details, see *Setting Naming Conventions* on page 541.

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (   OUT chSQLSTATE_OUT   CHAR(5),
                                  OUT intSQLCODE_OUT   INT,
                                  INOUT l_name char(20),
                                  INOUT f_name char(20))
    RESULT SETS 1
    LANGUAGE SQL
-------------------------------------------------------------------------
-- SQL Stored Procedure
-------------------------------------------------------------------------
P1: BEGIN
    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT
           EDAQA.NF29005.SSN5 AS SSN5,
           EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
           EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
           EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
           EDAQA.NF29005.SEX5 AS SEX5
        FROM
           EDAQA.NF29005
        WHERE
           (
             ( EDAQA.NF29005.LAST_NAME5 =  l_name )
        AND
             ( EDAQA.NF29005.FIRST_NAME5 = f_name )
           );
     -- Cursor left open for client application
    OPEN cursor1;
    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';
END P1   @
```

Chapter 20

Using the Adapter for DB Heritage Files

The Adapter for DB Heritage Files allows applications to access and read native IBM i data sources (also known as Physical, Logical, and Multi-format logical data base files) using the Open Query Facility (OPNQRYF). The adapter converts application requests into native OPNQRYF statements and returns optimized answer sets to the requesting application. All physical, logical, and multi-format formats are supported.

Note: In previous releases, this adapter was called DBFILE or DB File.

In this chapter:

Preparing the DB Heritage Files Environment

The Adapter for DB Heritage Files does not require setting any environment variables.

Configuring the Adapter for DB Heritage Files

You can configure the Adapter for DB Heritage Files from the Web Console or the Data Management Console.

Procedure: How to Configure the Adapter for DB Heritage Files

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Note: The *DB Heritage Files* adapter is under the *Sequential and Indexed* group folder. No input parameters are required.

Managing DB Heritage Files Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the DB Heritage Files data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each DB Heritage Files file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server metadata.

Note: Refer to *DB Heritage Standard Master File Attributes* on page 558, if you wish to review general information about Master File attributes, create a synonym based on attributes other than a standard DDS data definition, or make manual adjustments to synonyms generated using graphical tools.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Applications or Adapters pages of the Web Console.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Reference: Synonym Creation Parameters for DB Heritage Files

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select Files

Choose the type of file you wish to see from the following options: All, Physical, or Logical. (At this level, there is no differentiation between logical and multi-format logical.)

Library Name

Supply the name of the library in which the physical and/or logical files reside. (Wildcards are not permitted.)

Note: When you create a synonym for DB Heritage Files on the IBM i platform, standard IBM i naming conventions apply to the target data source. Therefore, the Adapter for DB Heritage Files supports the use of double-quotation marks around any library name and/or file name that contains lower case or NLS characters.

Filter by Name

If you wish to limit retrieval, enter a full file name or a partial name with a wildcard symbol % in the Filter by Name input box. A full name returns just that entry. A name with a wildcard symbol may return many entries.

Use text description as field name

Click this check box to use descriptive text from title fields as more meaningful field names in the synonym.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Select objects for synonym creation

To select all member names in the list, select the check box to the left of the *Default Synonym Name* column heading. (If the library contains a large number of files, this check box will not appear, however, synonyms will be created for all files automatically.)

To select specific member names, click the corresponding check boxes.

Not all listed objects are data-related. Be sure to select those that are appropriate for creating synonyms.

Note: Mixed-case names or names with NLS character will appear in double quotation marks. However, the double quotation marks will be replaced by underscore characters in the Default Synonym Name column.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Note: If you see a synonym name surrounded by underscore characters (as described in the previous note), you can remove the underscores if you wish.

Example: Creating a Synonym for a DB Heritage Files Data Source

In this example, the field attributes from the original file initialization are used in the new metadata. (For related information about file initialization, see *DB Heritage Standard Master File Attributes* on page 558.

An internal field (RECFORM) is also automatically added to reference a particular record's format. The RECFORM ALIAS always has a value of the name of the record format and is always added (even for single format files).

To generate the following synonym from the Create Synonym panes:

1. Specify *SHIPPING* in the Library Name field, then click *Submit*.
2. On the second Create Synonym pane, choose the member name *DBF2901* from the list of objects that comprise the library and click *Create Synonym*.

   The synonym is created and added under the specified application directory (baseapp is the default).

   A status window displays the message: All Synonyms Created Successfully
3. From the message window, click *Applications* on the menu bar.
4. Open the baseapp application folder in the navigation pane and right-click the synonym *DBF2901*.
5. Select *Edit as Text* from the drop-down menu to view the generated Master File. Your selections are reflected in the DATASET field, prefixed with QSYS:

   Generated Master File:

```
FILENAME=DBF2901, SUFFIX=DBFILE  ,
 DATASET=QSYS:SHIPPING/DBF2901, $
  SEGMENT=DBF2901, SEGTYPE=S0, $
    FIELDNAME=CCODE1, ALIAS=COUNTRY_COD1, USAGE=A5, ACTUAL=A5,
      TITLE='COUNTRY_COD1', DESCRIPTION='Country Code Number 1', $
    FIELDNAME=CNAME1, ALIAS=COUNTRY_NAM1, USAGE=P16, ACTUAL=P8,
      TITLE='COUNTRY_NAM1', DESCRIPTION='Country Name Number 1', $
    FIELDNAME=RECFORM, ALIAS=DBF2901, USAGE=A10, ACTUAL=A10, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

DB Heritage Standard Master File Attributes

Traditional IBM i applications, such as RPG, use Data Description Specification (DDS) files to initialize data files (also known as native database files) with header information about fields, data types, sizes and, optionally, aliases (ALIAS), column headings (COLHDG), descriptive comments (TEXT), and other information such as keys.

The features of the DDS have remained essentially stable despite the advent on IBM i of newer languages, such as C and COBOL, and their associated features. For example, while COBOL file descriptions may contain OCCURS statements, DDS does not support OCCURS clauses; for DDS purposes, separate fields must be used to handle this construct.

For DB Heritage Files, the IBM i DSPFFD command is used to extract the DDS information that is initialized into a data file at synonym-creation time, generating a complete set of metadata. However, in some instances, applications can redefine the use of the DDS information drastically, unbeknownst to the DSPFFD tool since the header information remains unchanged. This might occur, for example, in a COBOL application that has a common set of keys, part of which is a rectype that is used to redefine the remainder of the record into any number of layouts. The presence of the common set of keys suggests that the application is writable, when, in fact, DB Heritage Files is a Read-only adapter. While this situation does not generally occur, if it does you can avoid problems by following these guidelines to define proper metadata:

- Keep in mind that the Adapter for DB Heritage Files is *Read only*. Although some of the descriptive information may give the impression that describing keys is required and would provide Write capability, that is, in fact, *not* the case.
- Although the Adapter for DB Heritage Files is not a read/write adapter, you can use it in conjunction with the Adapter for Db2, which is a read/write adapter. Db2 itself provides an automatic feature for reading and writing native database files, with the exception of multi-format logical files, dynamically by using DSPFFD information on the fly. Thus, WebFOCUS can read and write DB Heritage Files data if the Adapter for Db2 is used to access the data. This two-adapter method enables you to take advantage of features of each one:
  - Adapter for Db2 read/write capabilities.
  - Adapter for DB Heritage Files support of Multi-Format Logical files and faster processing capabilities for large quantities of records.

  Tip: You can set up metadata for both methods and switch between them based upon the needs of your application.
- If a COBOL FD is available, you can use it with the Adapter for Flat Files to generate metadata with a SUFFIX=FIX value. You can then change the SUFFIX to DBFILE and add the DATASET= *value*.

For related information about DSPFFD, see *DSPFFD Behavior* on page 561.

Reference: Standard Master File Attributes for DB Heritage Files

Most standard Master File attributes are used with DB Heritage Files data sources in the standard way.

- **FILENAME.** The FILENAME attribute is the logical name of the file. It typically matches the Master File name.

- **SUFFIX.** The SUFFIX attribute in the declaration has the value DBFILE to indicate that access is for DB Heritage Files.
- **DATASET.** The DATASET attribute in the declaration indicates the default file to access. This attribute is optional and may be overridden by a FILEDEF command issued before a TABLE request (for example, in a focexec). The attribute value is preceded by the text *QSYS:*, which indicates that the file is a library (rather than a two-part application name). The syntax is:

  ```
  QSYS:[library/]file[(member)]
  ```

  If the library is not supplied, the user's library path is used to locate the file.
- **SEGNAME.** The SEGNAME attribute of the first or root segment in a Master File for a DB Heritage Files data source must be ROOT. The remaining segments can have any valid segment name.

  The only exception involves unrelated RECTYPEs, where the root SEGNAME must be DUMMY.

  All non-repeating data goes in the root segment. The remaining segments may have any valid name from one to eight characters.

  Any segment except the root is the descendant, or child, of another segment. The PARENT attribute supplies the name of the segment that is the hierarchical parent or owner of the current segment. If no PARENT attribute appears, the default is the immediately preceding segment. The PARENT name may be one to eight characters.
- **SEGTYPE.** The SEGTYPE attribute should be S0 for DB Heritage Files data sources.
- **GROUP.** The keys of a DB Heritage Files data source are defined in the segment declarations as GROUPs consisting of one or more fields.
- **FIELD.** The field names in the DB Heritage Files, along with the descriptive attributes, ALIAS, USAGE, and ACTUAL. For related information, see *USAGE and ACTUAL Values in DB Heritage Files* on page 560.

Reference: USAGE and ACTUAL Values in DB Heritage Files

Synonyms for DB Heritage Files will pick up certain additional data attributes which may have been declared in the DDS information that was used when the original data file was created.

For example, in a monetary declaration such as a dollar sign ($), if the IBM i system value for QCURSYN is the same as the symbol used in the DDS declaration, then the Master File *M* attribute (floating monetary symbol) is used. In this situation, the actual symbol that is displayed depends on the value defined by the SET CURRSYN command (dollar sign, by default), which may be set in an application or in a profile.

However, if the values for QCURSYN and the monetary sign in the DDS information do not match, but the DDS monetary value is defined as dollar, pound, euro, or yen, then !D, !L, !E, or !Y are displayed regardless of the value of CURRSYN on the assumption that the application is intentionally using multiple monetary symbols.

On the other hand, if QCURSYN and the monetary sign in the DDS information do not match and the DDS monetary value is not set to dollar, pound, euro, or yen, then the M symbol is displayed since dollar, pound, euro, and yen are the only explicitly supported characters for reports with mixed monetary symbols.

Among the other data attributes that may be picked up from the DDS information are date and time fields and the following symbols: *c* (suppresses commas), *C* (inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use), R (places CR after negative numbers).

Reference: DSPFFD Behavior

The DSPFFD field name value is used as the ALIAS=*value,* and the DSPFFD alias value is used as the FIELDNAME=*value*. This ensures that DSPFFD aliases, which can be too long to use as ALIAS values when creating synonyms in a hub/sub server configuration, are within the required character limit for hub/sub aliases.

While you may wish to refresh existing DB Heritage synonyms for other reasons, there is no explicit need to do so unless the underlying DSPFFD information actually changes. Existing DB Heritage related procedures will continue to run, with one possible difference. Report column titles, and possibly associated column widths, will display differently, if the request:

- Uses recreated metadata.
- Includes different values for FIELDNAME and ALIAS. (Frequently, these values are the same in the Master File.)
- Does not specify a TITLE value in the metadata.
- Does not assign explicit column titles.

Describing a Group Field

A single-segment data source may have only one key field, but it must still be described with a GROUP declaration. The group must have ALIAS=KEY.

Groups can also be assigned simply to provide convenient reference names for groups of fields. Suppose that you have a series of three fields for an employee: last name; first name; and middle initial. You use these three fields consistently to identify the employee. You can identify the three fields in your Master File as a GROUP named EMPINFO. Then, you can refer to these three linked fields as a single unit, called EMPINFO. When using the GROUP feature for non-keys, the parameter ALIAS= must still be used, but should not equal KEY.

For group fields, you must supply both the USAGE and ACTUAL formats in alphanumeric format. The length must be exactly the sum of the subordinate field lengths.

The GROUP declaration USAGE attribute specifies how many positions to use to describe the key in the data source. If a Master File does not completely describe the full key at least once, the following warning message appears:

```
(FOC1016) INVALID KEY DESCRIPTION IN MASTER FILE
```

The cluster key definition is compared to the Master File for length and displacement.

When you expand on the key in a RECTYPE data source, describe the key length in full on the last non-OCCURS segment on each data path.

Do not describe a group with ALIAS=KEY for OCCURS segments.

If the fields that make up a group key are not alphanumeric fields, the format of the group key is still alphanumeric, but its length is determined differently. The ACTUAL length is still the sum of the subordinate field lengths. However, the USAGE format is the sum of the internal storage lengths of the subordinate fields. You determine these internal storage lengths as follows:

- Fields of type I have a value of 4.
- Fields of type F have a value of 4.
- Fields of type P that are 8 bytes can have a USAGE of P15 or P16 (sign and decimal for a total of 15 digits). Fields that are 16 bytes have a USAGE of P17 or larger.
- Fields of type D have a value of 8.
- Alphanumeric fields have a value equal to the number of characters they contain as their field length.

Note:

- Since all group fields must be defined in alphanumeric format, those that include numeric component fields should not be used as verb objects in a report request.
- The MISSING attribute is not supported on the group field, but is supported on the individual fields comprising the group.

Syntax: How to Describe a DB Heritage Files Group Field

```
GROUP = keyname, ALIAS = KEY, USAGE = Ann, ACTUAL = Ann ,$
```

where:

keyname

Can have up to 66 characters.

Example: Describing a DB Heritage Files Group Field

In the library data source, the first field, PUBNO, can be described as a group key. The publisher's number consists of three elements: a number that identifies the publisher, one that identifies the author, and one that identifies the title. They can be described as a group key, consisting of a separate field for each element if the data source is a DB Heritage Files data structure. The Master File looks as follows:

```
FILE = LIBRARY5, SUFFIX = DBFILE,$
 SEGMENT = ROOT, SEGTYPE = S0,$
  GROUP = BOOKKEY      ,ALIAS = KEY ,USAGE = A10    ,ACTUAL = A10  ,$
   FIELDNAME = PUBNO   ,ALIAS = PN  ,USAGE = A3     ,ACTUAL = A3   ,$
   FIELDNAME = AUTHNO  ,ALIAS = AN  ,USAGE = A3     ,ACTUAL = A3   ,$
   FIELDNAME = TITLNO  ,ALIAS = TN  ,USAGE = A4     ,ACTUAL = A4   ,$
   FIELDNAME = AUTHOR   ,ALIAS = AT  ,USAGE = A25   ,ACTUAL = A25  ,$
   FIELDNAME = TITLE    ,ALIAS = TL  ,USAGE = A50   ,ACTUAL = A50  ,$
   FIELDNAME = BINDING  ,ALIAS = BI  ,USAGE = A1    ,ACTUAL = A1   ,$
   FIELDNAME = PRICE    ,ALIAS = PR  ,USAGE = D8.2N ,ACTUAL = D8   ,$
   FIELDNAME = SERIAL   ,ALIAS = SN  ,USAGE = A15   ,ACTUAL = A15  ,$
   FIELDNAME = SYNOPSIS ,ALIAS = SY  ,USAGE = A150  ,ACTUAL = A150 ,$
   FIELDNAME = RECTYPE  ,ALIAS = B   ,USAGE = A1    ,ACTUAL = A1   ,$
```

Example: Describing a DB Heritage Files Group Field With Multiple Formats

```
GROUP = A, ALIAS = KEY, USAGE = A14, ACTUAL = A8   ,$
 FIELDNAME = F1, ALIAS = F1, USAGE = P6, ACTUAL=P2 ,$
 FIELDNAME = F2, ALIAS = F2, USAGE = I9, ACTUAL=I4 ,$
 FIELDNAME = F3, ALIAS = F3, USAGE = A2, ACTUAL=A2 ,$
```

The lengths of the ACTUAL attributes for subordinate fields F1, F2, and F3 total 8, which is the length of the ACTUAL attribute of the group key. The display lengths of the USAGE attributes for the subordinate fields total 17. However, the length of the group key USAGE attribute is found by adding their internal storage lengths as specified by their field types: 8 for USAGE=P6, 4 for USAGE=I9, and 2 for USAGE=A2, for a total of 14.

Example: Accessing a Group Field With Multiple Formats

When you use a group field with multiple formats in a query, you must account for each position in the group, including trailing blanks or leading zeros. The following example illustrates how to access a group field with multiple formats in a query.

```
GROUP = GRPB, ALIAS = KEY, USAGE = A8, ACTUAL = A8 ,$
 FIELDNAME = FIELD1, ALIAS = F1, USAGE = A2, ACTUAL = A2 ,$
 FIELDNAME = FIELD2, ALIAS = F2, USAGE = I8, ACTUAL = I4 ,$
 FIELDNAME = FIELD3, ALIAS = F3, USAGE = A2, ACTUAL = A2 ,$
```

The values in fields F1 and F3 may include some trailing blanks, and the values in field F2 may include some leading zeros. When using the group in a query, you must account for each position. Because FIELD2 is a numeric field, you cannot specify the IF criteria as follows:

```
IF GRPB EQ 'A 0334BB'
```

You can eliminate this error by using a slash (/) to separate the components of the group key.

```
IF GRPB EQ 'A/334/BB'
```

Note: Blanks and leading zeros are assumed where needed to fill out the key.

Describe these multiply occurring fields by placing them in a separate segment. Fields A and B are placed in the root segment. Fields C1 and C2, which occur multiply in relation to A and B, are placed in a descendant segment. You use an additional segment attribute, the OCCURS attribute, to specify that these segments represent multiply occurring fields. In certain cases, you may also need a second attribute, called the POSITION attribute.

Using the OCCURS Attribute

The OCCURS attribute is an optional segment attribute used to describe records containing repeating fields or groups of fields. Define such records by describing the singly occurring fields in one segment, and the multiply occurring fields in a descendant segment. The OCCURS attribute appears in the declaration for the descendant segment.

You can have several sets of repeating fields in your data structure. Describe each of these sets of fields as a separate segment in your data source description. Sets of repeating fields can be divided into two basic types: parallel and nested.

Syntax: **How to Specify a Repeating Field**

```
OCCURS = occurstype
```

Possible values for *occurstype* are:

n

Is an integer value showing the number of occurrences (from 1 to 4095).

fieldname

Names a field in the parent segment whose integer value contains the number of occurrences of the descendant segment.

VARIABLE

Indicates that the number of occurrences varies from record to record. The number of occurrences is computed from the record length (for example, if the field lengths for the segment add up to 40, and 120 characters are read in, it means there are three occurrences).

Place the OCCURS attribute in your segment declaration after the PARENT attribute.

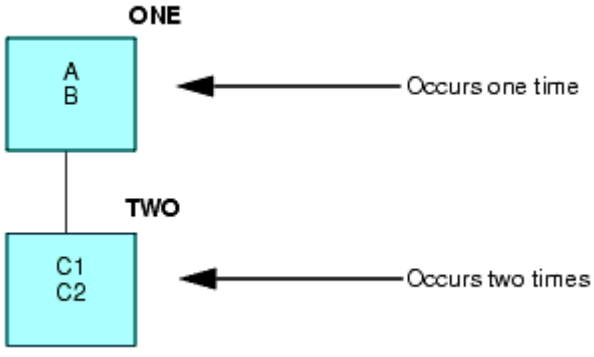
When different types of records are combined in one data source, each record type can contain only one segment defined as OCCURS=VARIABLE. It may have OCCURS descendants (if it contains a nested group), but it may not be followed by any other segment with the same parent—that is, there can be no other segments to its right in the hierarchical data structure. This restriction is necessary to ensure that data in the record is interpreted unambiguously.

Example: **Using the OCCURS Attribute**

Consider the following simple data structure:

A	B	C1	C2	C1	C2

You have two occurrences of fields C1 and C2 for every one occurrence of fields A and B. Thus, to describe this data source, you place fields A and B in the root segment, and fields C1 and C2 in a descendant segment, as shown here.



Describe this data source as follows:

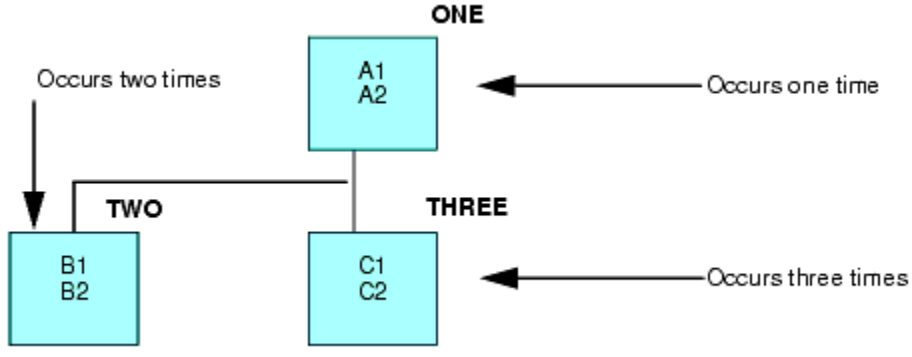
```
FILENAME = EXAMPLE1, SUFFIX = FIX, $
 SEGNAME = ONE, SEGTYPE=S0, $
  FIELDNAME = A,  ALIAS=, USAGE = A2, ACTUAL = A2, $
  FIELDNAME = B,  ALIAS=, USAGE = A1, ACTUAL = A1, $
 SEGNAME = TWO, PARENT = ONE, OCCURS = 2, SEGTYPE=S0, $
  FIELDNAME = C1, ALIAS=, USAGE = I4, ACTUAL = I2, $
  FIELDNAME = C2, ALIAS=, USAGE = I4, ACTUAL = I2, $
```

Describing a Parallel Set of Repeating Fields

Parallel sets of repeating fields are those that have nothing to do with one another (that is, they have no parent-child or logical relationship). Consider the following data structure:

A1	A2	B1	B2	B1	B2	C1	C2	C1	C2	C1	C2

In this example, fields B1 and B2 and fields C1 and C2 repeat within the record. The number of times that fields B1 and B2 occur has nothing to do with the number of times fields C1 and C2 occur. Fields B1 and B2 and fields C1 and C2 are parallel sets of repeating fields. They should be described in the data source description as children of the same parent, the segment that contains fields A1 and A2. The following data structure reflects their relationship.

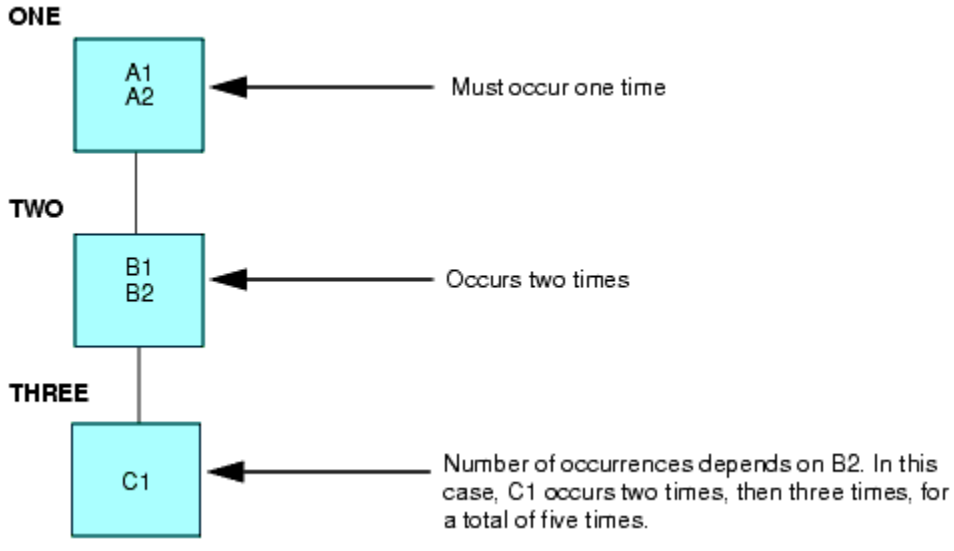


Describing a Nested Set of Repeating Fields

Nested sets of repeating fields are those whose occurrence depends on one another in some way. Consider the following data structure:

A1	A2	B1	B2	C1	C1	B1	B2	C1	C1	C1

In this example, field C1 only occurs after fields B1 and B2 occur once. It occurs varying numbers of times, recorded by a counter field, B2. There is not a set of occurrences of C1 which is not preceded by an occurrence of fields B1 and B2. Fields B1, B2, and C1 are a nested set of repeating fields. They can be represented by the following data structure.



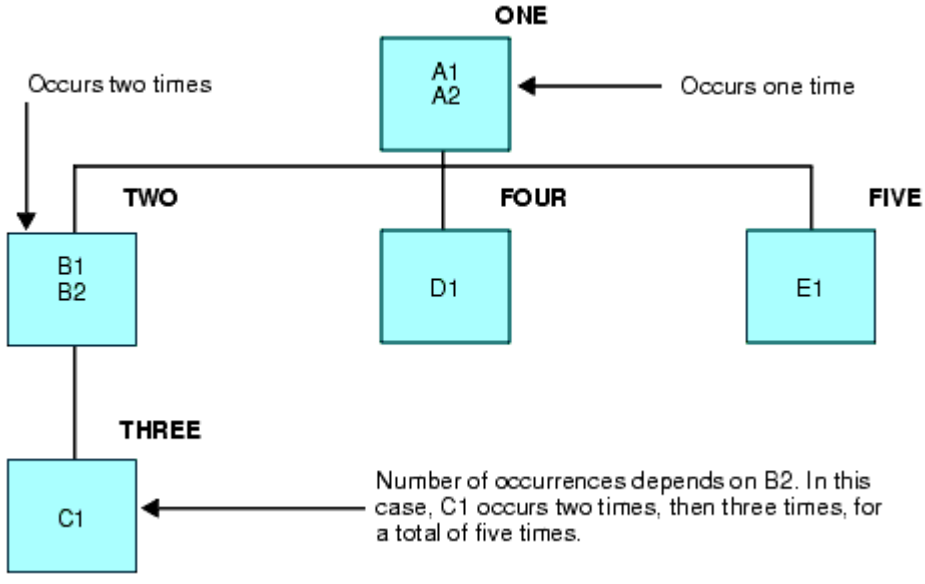
Since field C1 repeats with relation to fields B1 and B2, which repeat in relation to fields A1 and A2, field C1 is described as a separate, descendant segment of Segment TWO, which is in turn a descendant of Segment ONE.

Example: **Describing Parallel and Nested Repeating Fields**

The following data structure contains both nested and parallel sets of repeating fields.

A1	A2	B1	B2	C1	C1	C1	B1	B2	C1	C1	C1	C1	D1	D1	E1	E1	E1	E1

It produces the following data structure.



Describe this data source as follows. Notice that the assignment of the PARENT attributes shows you how the occurrences are nested.

```
FILENAME = EXAMPLE3, SUFFIX = FIX,$
 SEGNAME = ONE,    SEGTYPE=S0,$
  FIELDNAME = A1 ,ALIAS= ,ACTUAL = A1  ,USAGE = A1  ,$
  FIELDNAME = A2 ,ALIAS= ,ACTUAL = I1  ,USAGE = I1  ,$
 SEGNAME = TWO,    SEGTYPE=S0, PARENT = ONE, OCCURS = 2  ,$
  FIELDNAME = B1 ,ALIAS= ,ACTUAL = A15 ,USAGE = A15 ,$
  FIELDNAME = B2 ,ALIAS= ,ACTUAL = I1  ,USAGE = I1  ,$
 SEGNAME = THREE, SEGTYPE=S0, PARENT = TWO, OCCURS = B2 ,$
  FIELDNAME = C1 ,ALIAS= ,ACTUAL = A25 ,USAGE = A25 ,$
 SEGNAME = FOUR,  SEGTYPE=S0, PARENT = ONE, OCCURS = A2 ,$
  FIELDNAME = D1 ,ALIAS= ,ACTUAL = A15 ,USAGE = A15 ,$
 SEGNAME = FIVE,  SEGTYPE=S0, PARENT = ONE, OCCURS = VARIABLE,$
  FIELDNAME = E1 ,ALIAS= ,ACTUAL = A5  ,USAGE = A5  ,$
```

Note:

- Segments ONE, TWO, and THREE represent a nested group of repeating segments. Fields B1 and B2 occur a fixed number of times, so OCCURS equals 2. Field C1 occurs a certain number of times within each occurrence of fields B1 and B2. The number of times C1 occurs is determined by the value of field B2, which is a counter. In this case, its value is 3 for the first occurrence of Segment TWO and 4 for the second occurrence.

- Segments FOUR and FIVE consist of fields that repeat independently within the parent segment. They have no relationship to each other or to Segment TWO except their common parent, so they represent a parallel group of repeating segments.
- As in the case of Segment THREE, the number of times Segment FOUR occurs is determined by a counter in its parent, A2. In this case, the value of A2 is two.
- The number of times Segment FIVE occurs is variable. This means that all the rest of the fields in the record (all those to the right of the first occurrence of E1) are read as recurrences of field E1. To ensure that data in the record is interpreted unambiguously, a segment defined as OCCURS=VARIABLE must be at the end of the record. In a data structure diagram, it is the rightmost segment. Note that there can be only one segment defined as OCCURS=VARIABLE for each record type.

Using the POSITION Attribute

The POSITION attribute is an optional attribute used to describe a structure in which multiply occurring fields with an established number of occurrences are located in the middle of the record. You describe the data source as a hierarchical structure, made up of a parent segment and at least one child segment that contains the multiply occurring fields. The parent segment is made up of whatever singly occurring fields are in the record, as well as one or more alphanumeric fields that appear where the multiply occurring fields appear in the record. The alphanumeric field may be a dummy field that is the exact length of the combined multiply occurring fields. For example, if you have four occurrences of an eight-character field, the length of the field in the parent segment is 32 characters.

You can also use the POSITION attribute to re-describe fields with SEGTYPE=U. See *Redefining a Field in a DB Heritage Files Data Source* on page 572.

Syntax: How to Specify the Position of a Repeating Field

The POSITION attribute is described in the child segment. It gives the name of the field in the parent segment that specifies the starting position and overall length of the multiply occurring fields. The syntax of the POSITION attribute is

```
POSITION = fieldname
```

where:

fieldname

Is the name of the field in the parent segment that defines the starting position of the multiple field occurrences.

Example: **Specifying the Position of a Repeating Field**

Consider the following data structure:

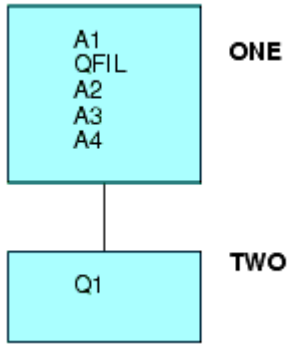
A1	Q1	Q1	Q1	Q1	A2	A3	A4

In this example, field Q1 repeats four times in the middle of the record. When you describe this structure, you specify a field or fields that occupy the position of the four Q1 fields in the record. You then assign the actual Q1 fields to a multiply occurring descendant segment. The POSITION attribute, specified in the descendant segment, gives the name of the field in the parent segment that identifies the starting position and overall length of the Q fields.

Use the following Master File to describe this structure.

```
FILENAME = EXAMPLE3, SUFFIX = FIX,$
 SEGNAME = ONE, SEGTYPE=S0,$
  FIELDNAME = A1    ,ALIAS= ,USAGE = A14 ,ACTUAL = A14 ,$
  FIELDNAME = QFIL  ,ALIAS= ,USAGE = A32 ,ACTUAL = A32 ,$
  FIELDNAME = A2    ,ALIAS= ,USAGE = I2  ,ACTUAL = I2  ,$
  FIELDNAME = A3    ,ALIAS= ,USAGE = A10 ,ACTUAL = A10 ,$
  FIELDNAME = A4    ,ALIAS= ,USAGE = A15 ,ACTUAL = A15 ,$
 SEGNAME = TWO, SEGTYPE=S0, PARENT = ONE, POSITION = QFIL, OCCURS = 4 ,$
  FIELDNAME = Q1    ,ALIAS= ,USAGE = D8  ,ACTUAL = D8  ,$
```

This produces the following structure:



If the total length of the multiply occurring fields is longer than 4095, you can use a filler field after the dummy field to make up the remaining length. This is required, because the format of an alphanumeric field cannot exceed 4095 bytes.

Notice that this structure works only if you have a fixed number of occurrences of the repeating field. This means the OCCURS attribute of the descendant segment must be of the type OCCURS=*n*. OCCURS=*fieldname* or OCCURS=VARIABLE does not work.

Specifying the ORDER Field

In an OCCURS segment, the order of the data may be significant. For example, the values may represent monthly or quarterly data, but the record itself may not explicitly specify the month or quarter to which the data applies.

To associate a sequence number with each occurrence of the field, you may define an internal counter field in any OCCURS segment. A value is automatically supplied that defines the sequence number of each repeating group.

Syntax: How to Specify the Sequence of a Repeating Field

The syntax rules for an ORDER field are:

- It must be the last field described in an OCCURS segment.
- The field name is arbitrary.
- The ALIAS is ORDER.
- The USAGE is I*n*, with any appropriate edit options.
- The ACTUAL is I4.

For example:

```
FIELD = ACT_MONTH, ALIAS = ORDER, USAGE = I2MT, ACTUAL = I4, $
```

Order values are 1, 2, 3, and so on, within each occurrence of the segment. The value is assigned prior to any selection tests that might accept or reject the record, and so it can be used in a selection test.

For example, to obtain data for only the month of June, type:

```
SUM AMOUNT...
WHERE ACT_MONTH IS 6
```

The ORDER field is a virtual field used internally. It does not alter the logical record length (LRECL) of the data source being accessed.

Redefining a Field in a DB Heritage Files Data Source

Redefining record fields in non-FOCUS data sources is supported. This enables you to describe a field with an alternate layout.

Within the Master File, the redefined fields must be described in a separate unique segment (SEGTYPE=U) using the POSITION=*fieldname* and OCCURS=1 attributes.

The redefined fields can have any user-defined name.

Syntax: How to Redefine a Field

```
SEGNAME = segname, SEGTYPE = U, PARENT = parentseg,
OCCURS = 1, POSITION = fieldname,$
```

where:

segname

Is the name of the segment.

parentseg

Is the name of the parent segment.

fieldname

Is the name of the first field being redefined. Using the unique segment with redefined fields helps avoid problems with multipath reporting.

A one-to-one relationship forms between the parent record and the redefined segment.

Example: Redefining a DB Heritage Files Structure

The following example illustrates redefinition of the DB Heritage Files structure described in the COBOL file description, where the COBOL FD is:

```
01 ALLFIELDS
  02 FLD1   PIC X(4)         -  this describes alpha/numeric data
  02 FLD2   PIC X(4)         -  this describes numeric data
  02 RFLD1  PIC 9(5)V99 COMP-3 REDEFINES FLD2
  02 FLD3   PIC X(8)         -  this describes alpha/numeric data

FILE = REDEF, SUFFIX = DBFILE,$
 SEGNAME = ONE, SEGTYPE = S0,$
  GROUP = RKEY, ALIAS = KEY, USAGE = A4 ,ACTUAL = A4 ,$
   FIELDNAME = FLD1,,  USAGE = A4   ,ACTUAL = A4 ,$
   FIELDNAME = FLD2,,  USAGE = A4   ,ACTUAL = A4 ,$
   FIELDNAME = FLD3,,  USAGE = A8   ,ACTUAL = A8 ,$
 SEGNAME = TWO, SEGTYPE = U, POSITION = FLD2, OCCURS = 1, PARENT = ONE ,$
   FIELDNAME = RFLD1,, USAGE = P8.2 ,ACTUAL = Z4 ,$
```

Reference: Special Considerations for Redefining a Field

- Redefinition is a read-only feature and is used only for presenting an alternate view of the data. It is not used for changing the format of the data.

- For non-alphanumeric fields, you must know your data. Attempts to print numeric fields that contain alphanumeric data produce data exceptions or errors converting values. It is recommended that the first definition always be alphanumeric to avoid conversion errors.
- More than one field can be redefined in a segment.
- Redefines are supported only for IDMS, IMS, VSAM, DB Heritage Files, Db2, and FIX data sources.

Extra-Large Record Length Support With DB Heritage Files

If a Master File describes a data source with records longer than 12K, which is typical for OCCURS segments and varchar fields, you must specify a larger record size in advance.

Syntax: How to Define the Maximum Record Length

```
SET MAXLRECL = nnnnn
```

where:

nnnnn

Is up to 32768.

For example, SET MAXLRECL=12000 allows handling of records that are 12000 bytes long. Once you have entered the SET MAXLRECL command, you can obtain the current value of the MAXLRECL parameter by using the ? SET MAXLRECL command.

If the actual record length is longer than specified, retrieval halts and the actual record length appears in hexadecimal notation.

Describing Multiple Record Types in DB Heritage Files

DB Heritage Files data sources can contain more than one type of record. When they do, they can be structured in one of two ways:

- A positional relationship may exist between the various record types, with a record of one type being followed by one or more records containing detailed information about the first record.

  If a positional relationship exists between the various record types, with a parent record of one type followed by one or more child records containing detail information about the parent, you describe the structure by defining the parent as the root, and the detail segments as descendants.

Some DB Heritage Files data sources are structured so that descendant records relate to each other through concatenating key fields. That is, the key fields of a parent record serve as the first part of the key of a child record. In such cases, the segment's key fields must be described using a GROUP declaration. Each segment's GROUP key fields consist of the renamed key fields from the parent segment plus the unique key field from the child record.

- The records have no meaningful positional relationship, and records of varying types exist independently of each other in the data source.

  If the records have no meaningful positional relationship, you have to provide some means for interpreting the type of record that has been read. Do this by creating a dummy root segment for the records.

Key-sequenced DB Heritage Files data sources also use the RECTYPE attribute to distinguish various record types within the data source.

A parent does not always share its RECTYPE with its descendants. It may share some other identifying piece of information, such as the PUBNO in the example. This is the field that should be included in the parent key, as well as all of its descendant keys, to relate them.

When using the RECTYPE attribute in DB Heritage Files data sources with group keys, the RECTYPE field can be part of the segment group key only when it belongs to a segment that has no descendants, or to a segment whose descendants are described with an OCCURS attribute. In *Describing Positionally Related Records* on page 577, the RECTYPE field is added to the group key in the SERIANO segment, the lowest descendant segment in the chain.

Describing a RECTYPE Field

When a data source contains multiple record types, there must be a field in the records themselves that can be used to differentiate between the record types. You can find information on this field in your existing description of the data source (for example, a COBOL FD statement). This field must appear in the same physical location of each record. For example, columns 79 and 80 can contain a different two-digit code for each unique record type. Describe this identifying field with the field name RECTYPE.

Another technique for redefining the parts of records is to use the MAPFIELD and MAPVALUE attributes described in *Describing a Repeating Group Using MAPFIELD* on page 591.

Syntax: How to Specify a Record Type Field

The RECTYPE field must fall in the same physical location of each record in the data source, or the record is ignored. The syntax to describe the RECTYPE field is

```
FIELDNAME = RECTYPE, ALIAS = value, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

value

Is the record type in alphanumeric format, if an ACCEPT list is not specified. If there is an ACCEPT list, this can be any value.

format

Is the data type of the field. In addition to RECTYPE fields in alphanumeric format, RECTYPE fields in packed and integer formats (formats P and I) are supported. Possible values are:

A*n* (where *n* is 1-4095) indicates character data, including letters, digits, and other characters.

I*n* indicates ACTUAL (internal) format binary integers:

- I1 = single-byte binary integer.
- I2 = half-word binary integer (2 bytes).
- I4 = full-word binary integer (4 bytes).

The USAGE format can be I1 through I9, depending on the magnitude of the ACTUAL format.

P*n* (where *n* is 1-16) indicates packed decimal ACTUAL (internal) format. n is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign. For example, P6 means 11 digits plus a sign.

If the field contains an assumed decimal point, represent the field with a USAGE format of P*m.n*, where *m* is the total number of digits, and *n* is the number of decimal places. Thus, P11.1 means an eleven-digit number with one decimal place.

list

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Separate each item in the list with either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks. The list may contain a single RECTYPE value. For example:

```
FIELDNAME = RECTYPE, ALIAS = TYPEABC, USAGE = A1,
ACTUAL = A1, ACCEPT = A OR B OR C, $
```

range

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed within single quotation marks.

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order. For example:

```
FIELDNAME = RECTYPE, ALIAS = ACCTREC, USAGE = P3,
ACTUAL = P2, ACCEPT = 100 TO 200, $
```

Example: Specifying the RECTYPE Field

The following field description is of a one-byte packed RECTYPE field containing the value 1:

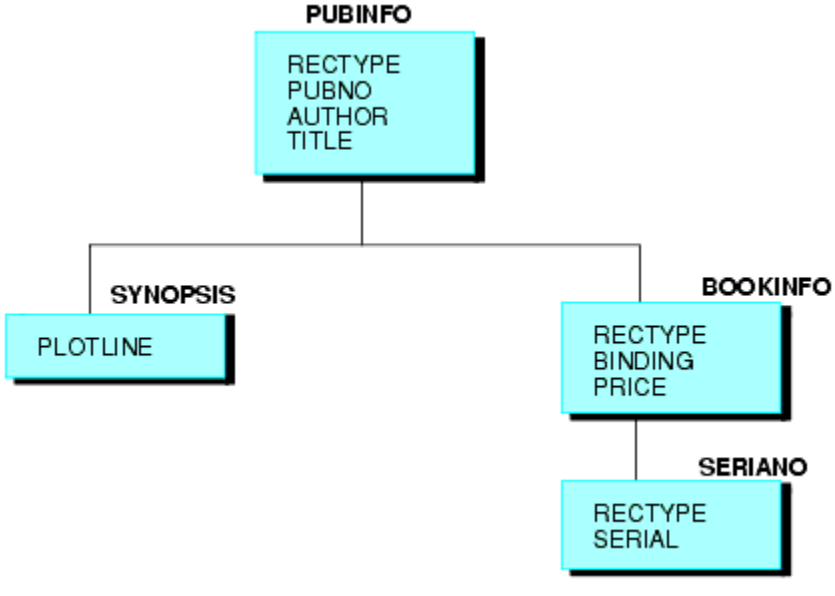
```
FIELD = RECTYPE, ALIAS = 1, USAGE = P1, ACTUAL = P1, $
```

The following field description is of a three-byte alphanumeric RECTYPE field containing the value A34:

```
FIELD = RECTYPE, ALIAS = A34, USAGE = A3, ACTUAL = A3,$
```

Describing Positionally Related Records

The following diagram shows a more complex version of the library data source.



Information that is common to all copies of a given book (the identifying number, the author name, and its title) has the same record type. All of this information is assigned to the root segment in the Master File. The synopsis is common to all copies of a given book, but in this data source it is described as a series of repeating fields of ten characters each, in order to save space.

The synopsis is assigned to its own subordinate segment with an attribute of OCCURS=VARIABLE in the Master File. Although there are segments in the diagram to the right of the OCCURS=VARIABLE segment, OCCURS=VARIABLE is the rightmost segment within its own record type. Only segments with a RECTYPE that is different from the OCCURS=VARIABLE segment can appear to its right in the structure. Note also that the OCCURS=VARIABLE segment does not have a RECTYPE. This is because it is part of the same record as its parent segment.

Binding and price can vary among copies of a given title. For instance, the library may have two different versions of *Pamela*, one a paperback costing $7.95, the other a hardcover costing $15.50. These two fields are of a second record type, and are assigned to a descendant segment in the Master File.

Finally, every copy of the book in the library has its own identifying serial number, which is described in a field of record type S. In the Master File, this information is assigned to a segment that is a child of the segment containing the binding and price information.

Use the following Master File to describe this data source:

```
FILENAME = LIBRARY2, SUFFIX = FIX,$
 SEGNAME = PUBINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = P     ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = PUBNO    ,ALIAS = PN    ,USAGE = A10   ,ACTUAL = A10 ,$
  FIELDNAME = AUTHOR   ,ALIAS = AT    ,USAGE = A25   ,ACTUAL = A25 ,$
  FIELDNAME = TITLE    ,ALIAS = TL    ,USAGE = A50   ,ACTUAL = A50 ,$
 SEGNAME = SYNOPSIS, PARENT = PUBINFO,  OCCURS = VARIABLE, SEGTYPE = S0,$
  FIELDNAME = PLOTLINE ,ALIAS = PLOTL ,USAGE = A10   ,ACTUAL = A10 ,$
 SEGNAME = BOOKINFO, PARENT = PUBINFO,  SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = B     ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = BINDING  ,ALIAS = BI    ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = PRICE    ,ALIAS = PR    ,USAGE = D8.2N ,ACTUAL = D8  ,$
 SEGNAME = SERIANO,  PARENT = BOOKINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = S     ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = SERIAL   ,ALIAS = SN    ,USAGE = A15   ,ACTUAL = A15 ,$
```

Note that each segment, except OCCURS, contains a field named RECTYPE and that the ALIAS for the field contains a unique value for each segment (P, B, and S). If there is a record in this data source with a RECTYPE other than P, B, or S, the record is ignored. The RECTYPE field must fall in the same physical location in each record.

Ordering of Records in the Data Source

Physical order determines parent/child relationships in sequential records. Every parent record does not need descendants. Specify how you want data in missing segment instances handled in your reports by using the SET command to change the ALL parameter.

In the example *Describing Positionally Related Records* on page 577, if the first record in the data source is not a PUBINFO record, the record is considered to be a child without a parent. Any information allotted to the SYNOPSIS segment appears in the PUBINFO record. The next record may be a BOOKINFO or even another PUBINFO (in which case the first PUBINFO is assumed to have no descendants). Any SERIANO records are assumed to be descendants of the previous BOOKINFO record. If a SERIANO record follows a PUBINFO record with no intervening BOOKINFO, it is treated as if it has no parent.

Example: Describing DB Heritage Files Positionally Related Records

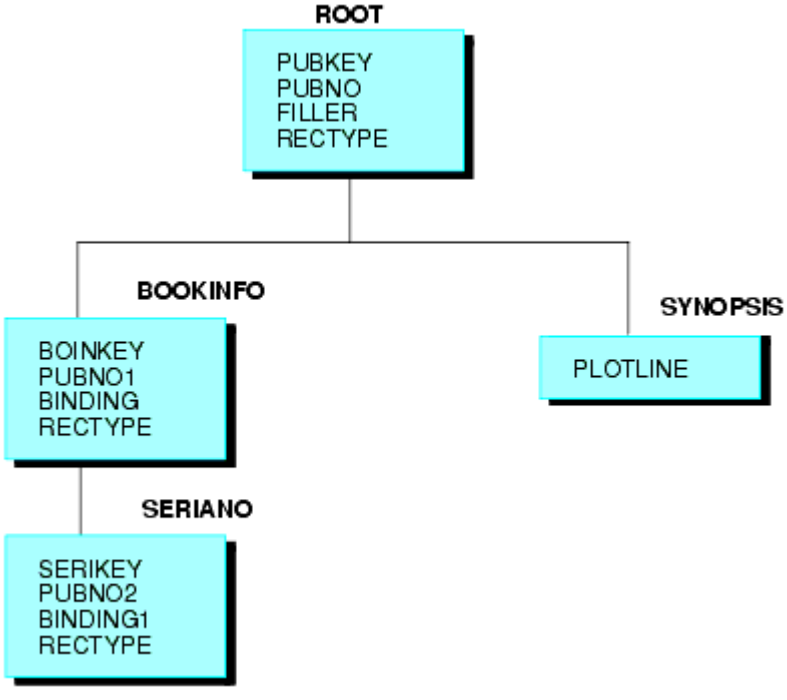
Consider the following DB Heritage Files data source that contains three types of records. The ROOT records have a key that consists of the publisher number, PUBNO. The BOOKINFO segment has a key that consists of that same publisher number, plus a hard- or soft-cover indicator, BINDING. The SERIANO segment key consists of the first two elements, plus a record type field, RECTYPE.

```
FILENAME = LIBRARY6, SUFFIX = DBFILE,$
 SEGNAME = ROOT, SEGTYPE = S0,$
  GROUP=PUBKEY         ,ALIAS=KEY   ,USAGE=A10   ,ACTUAL=A10  ,$
   FIELDNAME=PUBNO     ,ALIAS=PN    ,USAGE=A10   ,ACTUAL=A10  ,$
  FIELDNAME=FILLER     ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE    ,ALIAS=1     ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=AUTHOR     ,ALIAS=AT    ,USAGE=A25   ,ACTUAL=A25  ,$
  FIELDNAME=TITLE      ,ALIAS=TL    ,USAGE=A50   ,ACTUAL=A50  ,$
 SEGNAME=BOOKINFO,PARENT=ROOT,   SEGTYPE=S0,$
  GROUP=BOINKEY        ,ALIAS=KEY   ,USAGE=A11   ,ACTUAL=A11  ,$
   FIELDNAME=PUBNO1    ,ALIAS=P1    ,USAGE=A10   ,ACTUAL=A10  ,$
   FIELDNAME=BINDING   ,ALIAS=BI    ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE    ,ALIAS=2     ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=PRICE      ,ALIAS=PR    ,USAGE=D8.2N ,ACTUAL=D8   ,$
 SEGNAME=SERIANO, PARENT=BOOKINFO,SEGTYPE=S0,$
  GROUP=SERIKEY        ,ALIAS=KEY   ,USAGE=A12   ,ACTUAL=A12  ,$
   FIELDNAME=PUBNO2    ,ALIAS=P2    ,USAGE=A10   ,ACTUAL=A10  ,$
   FIELDNAME=BINDING1  ,ALIAS=B1    ,USAGE=A1    ,ACTUAL=A1   ,$
   FIELDNAME=RECTYPE   ,ALIAS=3     ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=SERIAL     ,ALIAS=SN    ,USAGE=A15   ,ACTUAL=A15  ,$
 SEGNAME=SYNOPSIS,PARENT=ROOT,    SEGTYPE=S0, OCCURS=VARIABLE,$
  FIELDNAME=PLOTLINE   ,ALIAS=PLOTL ,USAGE=A10   ,ACTUAL=A10  ,$
```

Notice that the length of the key fields specified in the USAGE and ACTUAL attributes of a GROUP declaration is the length of the key fields from the parent segments, plus the length of the added field of the child segment (RECTYPE field). In the example above, the length of the GROUP key SERIKEY equals the length of PUBNO2 and BINDING1, the group key from the parent segment, plus the length of RECTYPE, the field added to the group key in the child segment. The length of the key increases as you traverse the structure.

Note: Each segment key describes as much of the true key as needed to find the next instance of that segment.

In the sample data source, the repetition of the publisher's number as PUBNO1 and PUBNO2 in the descendant segments interrelates the three types of records. The data source can be diagrammed as the following structure:



A typical query may request information on price and call numbers for a specific publisher number:

```
PRINT PRICE AND SERIAL BY PUBNO
IF PUBNO EQ 1234567890 OR 9876054321
```

Since PUBNO is part of the key, retrieval can occur quickly, and the processing continues. To further speed retrieval, add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Records

Some data sources do not have records that are related to one another. That is, the key of one record type is independent of the keys of other record types. To describe data sources with unrelated records, define a dummy root segment for the record types. The following rules apply to the dummy root segment:

- The name of the root segment must be DUMMY.
- It must have only one field with a blank name and alias.
- The USAGE and ACTUAL attributes must both be A1.

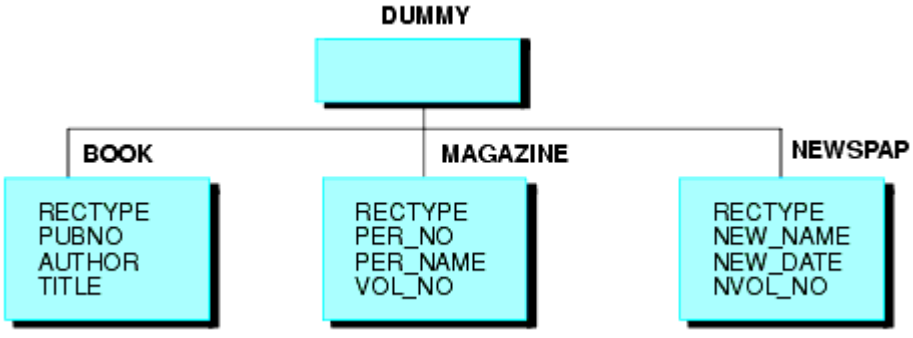
All other non-repeating segments must point to the dummy root as their parent. Except for the root, all non-repeating segments must have a RECTYPE and a PARENT attribute and describe the full key. If the data source does not have a key, the group should not be described. RECTYPEs may be anywhere in the record.

Example: Describing Unrelated Records Using a Dummy Root Segment

The library data source has three types of records: book information, magazine information, and newspaper information. Since these three record types have nothing in common, they cannot be described as parent records followed by detail records.

The data source can look like this:

A structure such as the following can also describe this data source.



The Master File for the structure in this example is:

```
FILENAME = LIBRARY3, SUFFIX = FIX,$
 SEGMENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME=           ,ALIAS=     ,USAGE = A1    ,ACTUAL = A1   ,$
 SEGMENT = BOOK, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = B  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PUBNO    ,ALIAS = PN ,USAGE = A10   ,ACTUAL = A10  ,$
  FIELDNAME = AUTHOR   ,ALIAS = AT ,USAGE = A25   ,ACTUAL = A25  ,$
  FIELDNAME = TITLE    ,ALIAS = TL ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = BINDING  ,ALIAS = BI ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PRICE    ,ALIAS = PR ,USAGE = D8.2N ,ACTUAL = D8   ,$
  FIELDNAME = SERIAL   ,ALIAS = SN ,USAGE = A15   ,ACTUAL = A15  ,$
  FIELDNAME = SYNOPSIS ,ALIAS = SY ,USAGE = A150  ,ACTUAL = A150 ,$
 SEGMENT = MAGAZINE, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = M  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PER_NO   ,ALIAS = PN ,USAGE = A10   ,ACTUAL = A10  ,$
  FIELDNAME = PER_NAME ,ALIAS = NA ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = VOL_NO   ,ALIAS = VN ,USAGE = I2    ,ACTUAL = I2   ,$
  FIELDNAME = ISSUE_NO ,ALIAS = IN ,USAGE = I2    ,ACTUAL = I2   ,$
  FIELDNAME = PER_DATE ,ALIAS = DT ,USAGE = I6MDY ,ACTUAL = I6   ,$
 SEGMENT = NEWSPAP, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = N  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = NEW_NAME ,ALIAS = NN ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = NEW_DATE ,ALIAS = ND ,USAGE = I6MDY ,ACTUAL = I6   ,$
  FIELDNAME = NVOL_NO  ,ALIAS = NV ,USAGE = I2    ,ACTUAL = I2   ,$
  FIELDNAME = ISSUE    ,ALIAS = NI ,USAGE = I2    ,ACTUAL = I2   ,$
```

Example: Describing a DB Heritage Files Data Source With Unrelated Records

Consider another data source containing information on the library. This data source has three types of records: book information, magazine information, and newspaper information.

There are two possible structures:

The RECTYPE is the beginning of the key. The key structure is:

RECTYPE B	Book Code
RECTYPE M	Magazine Code
RECTYPE N	Newspaper Code

The sequence of records is:

Book
Book
Magazine
Magazine
Newspaper
Newspaper

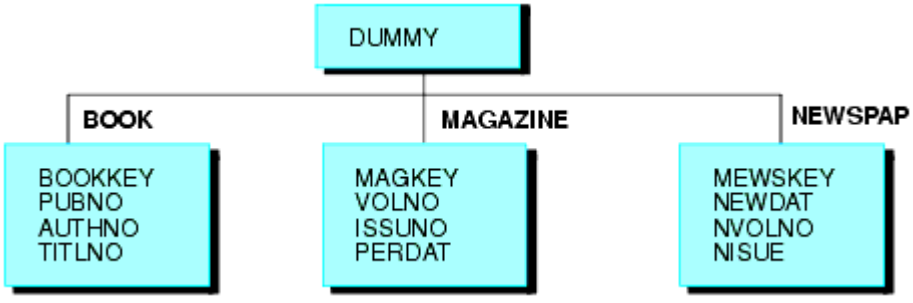
Note the difference between the use of the RECTYPE here and its use when the records are positionally related. In this case, the codes are unrelated and the database designer has chosen to accumulate the records by type first (all the book information together, all the magazine information together, and all the newspaper information together), so the RECTYPE may be the initial part of the key.

The RECTYPE is not in the beginning of the key or is outside of the key. The key structure is:

Book Code
Magazine Code
Newspaper Code

The sequence of record types in the data source can be arbitrary.

Both types of file structure can be represented by the following:



Example: **Describing a Key and a Record Type for a DB Heritage Files Data Source With Unrelated Records**

```
FILE=LIBRARY7, SUFFIX=DBFILE,$
 SEGMENT=DUMMY,$
  FIELDNAME=          ,ALIAS=    ,USAGE=A1     ,ACTUAL=A1   ,$
 SEGMENT=BOOK,      PARENT=DUMMY,SEGTYPE=S0,$
  GROUP=BOOKKEY       ,ALIAS=KEY ,USAGE=A11    ,ACTUAL=A11  ,$
   FIELDNAME=PUBNO    ,ALIAS=PN  ,USAGE=A3     ,ACTUAL=A3   ,$
   FIELDNAME=AUTHNO   ,ALIAS=AN  ,USAGE=A3     ,ACTUAL=A3   ,$
   FIELDNAME=TITLNO   ,ALIAS=TN  ,USAGE=A4     ,ACTUAL=A4   ,$
   FIELDNAME=RECTYPE  ,ALIAS=B   ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=AUTHOR    ,ALIAS=AT  ,USAGE=A25    ,ACTUAL=A25  ,$
  FIELDNAME=TITLE     ,ALIAS=TL  ,USAGE=A50    ,ACTUAL=A50  ,$
  FIELDNAME=BINDING   ,ALIAS=BI  ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=PRICE     ,ALIAS=PR  ,USAGE=D8.2N  ,ACTUAL=D8   ,$
  FIELDNAME=SERIAL    ,ALIAS=SN  ,USAGE=A15    ,ACTUAL=A15  ,$
  FIELDNAME=SYNOPSIS  ,ALIAS=SY  ,USAGE=A150   ,ACTUAL=A150 ,$
 SEGMENT=MAGAZINE,  PARENT=DUMMY,  SEGTYPE=S0,$
  GROUP=MAGKEY        ,ALIAS=KEY ,USAGE=A11    ,ACTUAL=A11  ,$
   FIELDNAME=VOLNO    ,ALIAS=VN  ,USAGE=A2     ,ACTUAL=A2   ,$
   FIELDNAME=ISSUNO   ,ALIAS=IN  ,USAGE=A2     ,ACTUAL=A2   ,$
   FIELDNAME=PERDAT   ,ALIAS=DT  ,USAGE=A6     ,ACTUAL=A6   ,$
   FIELDNAME=RECTYPE  ,ALIAS=M   ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=PER_NAME  ,ALIAS=PRN ,USAGE=A50    ,ACTUAL=A50  ,$
 SEGMENT=NEWSPAP,   PARENT=DUMMY,  SEGTYPE=S0,$
  GROUP=NEWSKEY       ,ALIAS=KEY ,USAGE=A11    ,ACTUAL=A11  ,$
   FIELDNAME=NEWDAT   ,ALIAS=ND  ,USAGE=A6     ,ACTUAL=A6   ,$
   FIELDNAME=NVOLNO   ,ALIAS=NV  ,USAGE=A2     ,ACTUAL=A2   ,$
   FIELDNAME=NISSUE   ,ALIAS=NI  ,USAGE=A2     ,ACTUAL=A2   ,$
   FIELDNAME=RECTYPE  ,ALIAS=N   ,USAGE=A1     ,ACTUAL=A1   ,$
  FIELDNAME=NEWNAME   ,ALIAS=NN  ,USAGE=A50    ,ACTUAL=A50  ,$
```

Using a Generalized Record Type

If your data source has multiple record types that share the same layout, you can specify a single generalized segment that describes all record types that have the common layout. By using a generalized segment, also known as a generalized RECTYPE, instead of one segment per record type, you reduce the number of segments you need to describe in the Master File.

When using a generalized segment, you identify RECTYPE values using the ACCEPT attribute. You can assign any value to the ALIAS attribute.

Syntax: How to Specify a Generalized Record Type

```
FIELDNAME = RECTYPE, ALIAS = alias, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

`RECTYPE`

Is the required field name.

Note: Since the field name, RECTYPE, may not be unique across segments, you should not use it in this way unless you qualify it. An alias is not required; you may leave it blank.

`alias`

Is any valid alias specification. You can specify a unique name as the alias value for the RECTYPE field only if you use the ACCEPT attribute. The alias can then be used in a TABLE request as a display field, a sort field, or in selection tests using either WHERE or IF.

`list`

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks. The list may contain a single RECTYPE value. For example:

```
FIELDNAME = RECTYPE, ALIAS = TYPEABC, USAGE = A1,
ACTUAL = A1, ACCEPT = A OR B OR C, $
```

`range`

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed within single quotation marks.

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order. For example:

```
FIELDNAME = RECTYPE, ALIAS = ACCTREC, USAGE = P3,
ACTUAL = P2, ACCEPT = 100 TO 200, $
```

Example: Using a Generalized Record Type

To illustrate the use of the generalized record type in a Master File, consider the following record layouts in the DOC data source. Record type DN is the root segment and contains the document number and title. Record types M, I, and C contain information about manuals, installation guides, and course guides, respectively. Notice that record types M and I have the same layout.

Record type DN:

```
---KEY---
+-------------------------------------------------------------------------+
DOCID   FILLER         RECTYPE     TITLE
+-------------------------------------------------------------------------+
```

Record type M:

```
--------KEY--------
+-------------------------------------------------------------------------+
MDOCID   MDATE         RECTYPE     MRELEASE          MPAGES     FILLER
+-------------------------------------------------------------------------+
```

Record type I:

```
--------KEY--------
+-------------------------------------------------------------------------+
IDOCID   IDATE         RECTYPE     IRELEASE          IPAGES     FILLER
+-------------------------------------------------------------------------+
```

Record type C:

```
--------KEY--------
+-------------------------------------------------------------------------+
CRSEDOC   CDATE        RECTYPE     COURSENUM  LEVEL  CPAGES     FILLER
+-------------------------------------------------------------------------+
```

Without the ACCEPT attribute, each of the four record types must be described as separate segments in the Master File. A unique set of field names must be provided for record type M and record type I, although they have the same layout.

The generalized RECTYPE capability enables you to code just one set of field names that applies to the record layout for both record type M and I. The ACCEPT attribute can be used for any RECTYPE specification, even when there is only one acceptable value.

```
FILENAME=DOC2, SUFFIX=DBFILE,$
SEGNAME=ROOT, SEGTYPE=SO,$
 GROUP=DOCNUM,     ALIAS=KEY,        A5,  A5,  $
  FIELD=DOCID,     ALIAS=SEQNUM,     A5,  A5,  $
 FIELD=FILLER,     ALIAS,=           A5,  A5,  $
 FIELD=RECTYPE,    ALIAS=DOCRECORD, A3,  A3,  ACCEPT = DN,$
 FIELD=TITLE,      ALIAS=,           A18, A18, $
SEGNAME=MANUALS, PARENT=ROOT, SEGTYPE=SO,$
 GROUP=MDOCNUM,    ALIAS=KEY, A10, A10,$
  FIELD=MDOCID,    ALIAS=MSEQNUM,    A5,  A5,  $
  FIELD=MDATE,     ALIAS=MPUBDATE,   A5,  A5,  $
 FIELD=RECTYPE,    ALIAS=MANUAL,     A3,  A3,  ACCEPT = M OR I,$
 FIELD=MRELEASE,   ALIAS=,           A7,  A7,  $
 FIELD=MPAGES,     ALIAS=,           I5,  A5,  $
 FIELD=FILLER,     ALIAS=,           A6,  A6,  $
SEGNAME=COURSES, PARENT=ROOT, SEGTYPE=SO,$
 GROUP=CRSEDOC,    ALIAS=KEY, A10, A10,$
  FIELD=CDOCID,    ALIAS=CSEQNUM,    A5,  A5,  $
  FIELD=CDATE,     ALIAS=CPUBDATE,   A5,  A5,  $
 FIELD=RECTYPE,    ALIAS=COURSE,     A3,  A3,  ACCEPT = C,$
 FIELD=COURSENUM, ALIAS=CNUM,       A4,  A4,  $
 FIELD=LEVEL,      ALIAS=,           A2,  A2,  $
 FIELD=CPAGES,     ALIAS=,           I5,  A5,  $
 FIELD=FILLER,     ALIAS=,           A7,  A7,  $
```

Combining Multiply-Occurring Fields and Multiple Record Types in DB Heritage Files

You can have two types of descendant segments in a single fixed-format DB Heritage Files data source:

- Descendant segments consisting of multiply occurring fields.
- Additional descendant segments consisting of multiple record types.

Describing a Multiply Occurring Field and Multiple Record Types

In the data structure shown below, the first record, of type 01, contains several different sequences of repeating fields, all of which must be described as descendant segments with an OCCURS attribute. The data source also contains two separate records, of types 02 and 03, which contain information that is related to that in record type 01.

The relationship between the records of various types is expressed as parent-child relationships. The children that contain record types 02 and 03 do not have an OCCURS attribute. They are distinguished from their parent by the field declaration where field=RECTYPE.

01	T1	N1	B1	B2	C1	C1	C1	D1	D1	D1	D1	D1	D1	D1	B1	B2	C1	C1	D1	D1	D1	D1	D1	D1

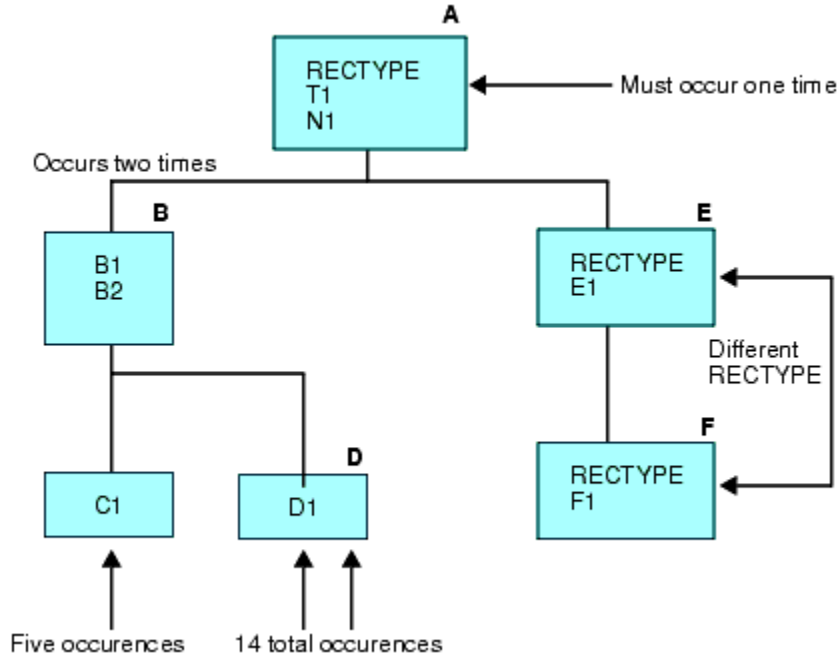
02	E1

03	F1

The description for this data source is:

```
FILENAME = EXAMPLE1, SUFFIX = FIX,$
 SEGNAME = A, SEGTYPE=S0,$
  FIELDNAME = RECTYPE   ,ALIAS = 01   ,USAGE = A2   ,ACTUAL = A2 ,$
  FIELDNAME = T1        ,ALIAS =      ,USAGE = A2   ,ACTUAL = A1 ,$
  FIELDNAME = N1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = B, PARENT = A, OCCURS = VARIABLE, SEGTYPE=S0,$
  FIELDNAME = B1        ,ALIAS =      ,USAGE = I2   ,ACTUAL = I2 ,$
  FIELDNAME = B2        ,ALIAS =      ,USAGE = I2   ,ACTUAL = I2 ,$
 SEGNAME = C, PARENT = B, OCCURS = B1, SEGTYPE=S0,$
  FIELDNAME = C1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = D, PARENT = B, OCCURS = 7,  SEGTYPE=S0,$
  FIELDNAME = D1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = E, PARENT = A, SEGTYPE=S0,$
  FIELDNAME = RECTYPE   ,ALIAS = 02   ,USAGE = A2   ,ACTUAL = A2 ,$
  FIELDNAME = E1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = F, PARENT = E, SEGTYPE=S0,$
  FIELDNAME = RECTYPE   ,ALIAS = 03   ,USAGE = A2   ,ACTUAL = A2 ,$
  FIELDNAME = F1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
```

It produces the following data structure:



Segments A, B, C, and D all belong to the same record type. Segments E and F each are stored as separate record types.

Note:

- Segments A, E, and F are different records that are related through their record types. The record type attribute consists of certain prescribed values, and is stored in a fixed location in the records. Records are expected to be retrieved in a given order. If the first record does not have a RECTYPE of 01, the record is considered to be a child without a parent. The next record can have a RECTYPE of either 01 (in which case, the first record is considered to have no descendants except the OCCURS descendants) or 02. A record with a RECTYPE of 03 can follow only a record with a RECTYPE of 02 (its parent) or another 03.
- The OCCURS descendants all belong to the record whose RECTYPE is 01. (This is not a necessary condition; records of any type can have OCCURS descendants.) Note that the OCCURS=VARIABLE segment, Segment B, is the rightmost segment within its own record type. If you look at the data structure, the pattern that makes up Segment B and its descendants (the repetition of fields B1, B2, C1, and D1) extends from the first mention of fields B1 and B2 to the end of the record.

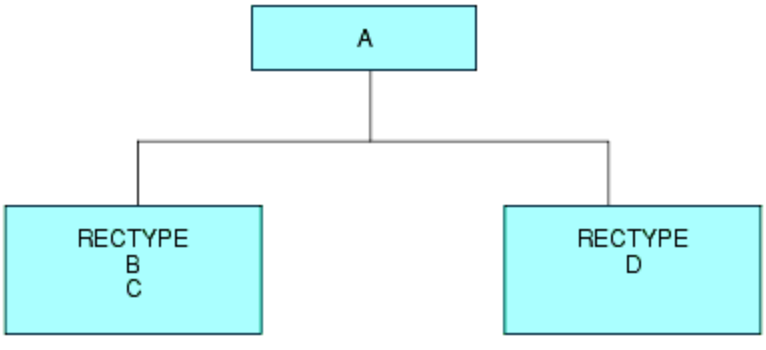
- Although fields C1 and D1 appear in separate segments, they are actually part of the repeating pattern that makes up the OCCURS=VARIABLE segment. Since they occur multiple times within Segment B, they are each assigned to their own descendant segment. The number of times field C1 occurs depends on the value of field B2. In the example, the first value of field B2 is 3; the second, 2. Field D1 occurs a fixed number of times, 7.

Describing a Repeating Group With RECTYPEs

Suppose you want to describe a data source that, schematically, looks like this:

```
A     RECTYPE  B C    RECTYPE   B C
A     RECTYPE  D      RECTYPE   D
```

You need to describe three segments in your Master File, with A as the root segment, and segments for B, C, and D as two descendant OCCURS segments for A.



Each of the two descendant OCCURS segments in this example depends on the RECTYPE indicator that appears for each occurrence.

All the rules of syntax for using RECTYPE fields and OCCURS segments also apply to RECTYPEs within OCCURS segments.

Since each OCCURS segment depends on the RECTYPE indicator for its evaluation, the RECTYPE must appear at the start of the OCCURS segment. This enables you to describe complex data sources, including those with nested and parallel repeating groups that depend on RECTYPEs.

Example: Describing a Repeating Group With RECTYPEs

In this example, B/C, and D represent a nested repeating group, and E represents a parallel repeating group.

A	RECTYPE B C	RECTYPE D	RECTYPE E	RECTYPE E

```
FILENAME=SAMPLE,SUFFIX=DBFILE,$
 SEGNAME=ROOT,SEGTYPE=S0,$
  GROUP=GRPKEY      ,ALIAS=KEY ,USAGE=A8 ,ACTUAL=A8 ,$
   FIELD=FLD000     ,E00       ,A08      ,A08       ,$
   FIELD=A_DATA     ,E01       ,A02      ,A02       ,$
 SEGNAME=SEG001,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD=RECTYPE     ,A01       ,A01      ,ACCEPT=B OR C ,$
  FIELD=B_OR_C_DATA ,E02       ,A08      ,A08       ,$
 SEGNAME=SEG002,PARENT=SEG001,OCCURS=VARIABLE,SEGTYPE=S0,$
  FIELD=RECTYPE     ,D         ,A01      ,A01       ,$
  FIELD=D_DATA      ,E03       ,A07      ,A07       ,$
 SEGNAME=SEG003,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD=RECTYPE     ,E         ,A01      ,A01       ,$
  FIELD=E_DATA      ,E04       ,A06      ,A06       ,$
```

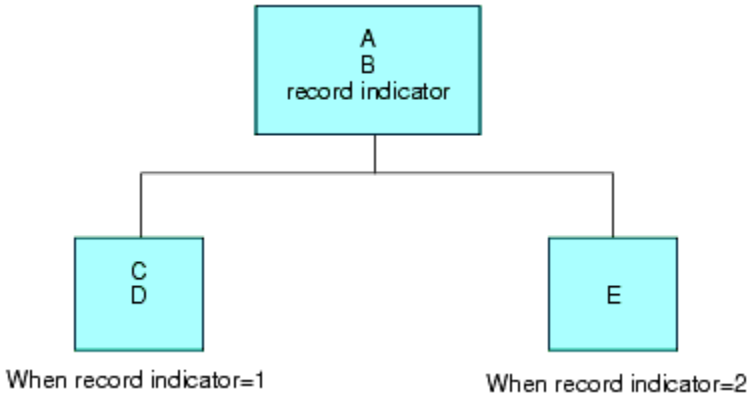
Describing a Repeating Group Using MAPFIELD

In another combination of record indicator and OCCURS, a record contains a record indicator that is followed by a repeating group. In this case, the record indicator is in the fixed portion of the record, not in each occurrence. Schematically, the record appears like this:

A B	record indicator (1)	C D	C D	C D
A B	record indicator (2)	E E		

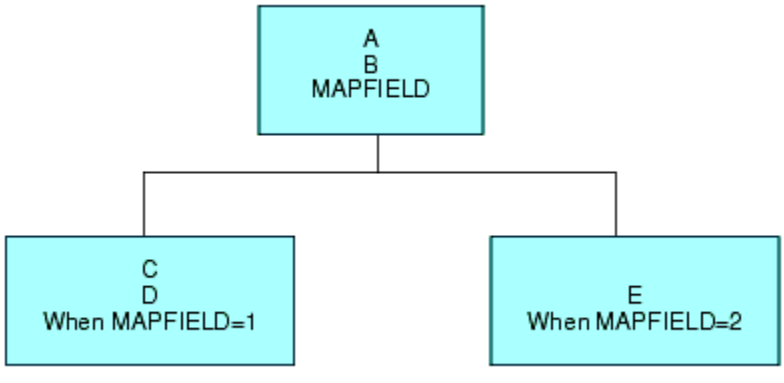
The first record contains header information, values for A and B, followed by an OCCURS segment of C and D that was identified by its preceding record indicator. The second record has a different record indicator and contains a different repeating group, this time for E.

The following diagram illustrates this relationship.



Since the OCCURS segments are identified by the record indicator rather than the parent A/B segment, you must use the keyword MAPFIELD. MAPFIELD identifies a field in the same way as RECTYPE, but since each OCCURS segments has its own value for MAPFIELD, the value of MAPFIELD is associated with each OCCURS segment by means of a complementary field named MAPVALUE.

The following diagram illustrates this relationship.



MAPFIELD is assigned as the ALIAS of the field that is the record indicator. It may have any name.

Syntax: How to Describe a Repeating Group With MAPFIELD

```
FIELD = name, ALIAS = MAPFIELD, USAGE = format, ACTUAL = format,$
```

where:

name

Is the name you choose to provide for this field.

ALIAS

MAPFIELD is assigned as the alias of the field that is the RECTYPE indicator.

USAGE

Follows the usual field format.

ACTUAL

Follows the usual field format.

The descendant segment values depend on the value of the MAPFIELD. They are described as separate segments, one for each possible value of MAPFIELD, and all descending from the segment that has the MAPFIELD. A special field, MAPVALUE, is described as the last field in these descendant segments after the ORDER field, if one has been used. The actual MAPFIELD value is supplied as the ALIAS of MAPVALUE.

Syntax: How to Use MAPFIELD for a Descendant Repeating Segment in a Repeating Group

```
FIELD = MAPVALUE, ALIAS = alias, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

MAPVALUE

Indicates that the segment depends on a MAPFIELD in its parent segment.

alias

Is the primary MAPFIELD value, if an ACCEPT list is not specified. If there is an ACCEPT list, this can be any value.

USAGE

Is the same format as the MAPFIELD format in the parent segment.

ACTUAL

Is the same format as the MAPFIELD format in the parent segment.

list

Is the list of one or more lines of specified MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single MAPFIELD value.

For example:

```
FIELDNAME = MAPFIELD, ALIAS = A, USAGE = A1, ACTUAL = A1,
ACCEPT = A OR B OR C,$
```

range

Is a range of one or more lines of MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

Example: Using MAPFIELD and MAPVALUE

Using the sample data source at the beginning of this section, the Master File for this data source looks like this:

```
FILENAME=EXAMPLE,SUFFIX=FIX,$
 SEGNAME=ROOT,SEGTYPE=S0,$
  FIELD =A,               ,A14   ,A14 ,$
  FIELD =B,               ,A10   ,A10 ,$
  FIELD =FLAG ,MAPFIELD   ,A01   ,A01 ,$
 SEGNAME=SEG001,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD =C,               ,A05   ,A05 ,$
  FIELD =D,               ,A07   ,A07 ,$
  FIELD =MAPVALUE ,1      ,A01   ,A01 ,$
 SEGNAME=SEG002,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD =E,               ,D12.2 ,D8  ,$
  FIELD =MAPVALUE ,2      ,A01   ,A01 ,$
```

Note: MAPFIELD can only exist on an OCCURS segment that has not been remapped. This means that the segment definition cannot contain POSITION=*fieldname*.

MAPFIELD and MAPVALUE may be used with SUFFIX=FIX and SUFFIX=DBFILE data sources.

Multi-Format Logical Files

Multi-format logical files are an accumulation of different record formats, each one from a separate physical file. Multi-format logical files enable placement of multiple physical files in one logical (hierarchical) file structure. Each of the physical files should have common keys.

Example: Associating a Logical File With Three Physical Files

The following are IBM i Data Description Specifications (DDSs) for three physical files and the logical file that associates them:

```
EMPLOYEE Physical File DDS

0001.00      A                                      UNIQUE
0002.00      A          R EMPINFOR
0003.00      A            EMP_ID         9A
0004.00      A            LAST_NAME     10A
0005.00      A            FIRST_NAME    10A
0006.00      A            ADDRESS       40A
0007.00      A          K EMP_ID

PENSION Physical File DDS

0001.00      A                                      UNIQUE
0002.00      A          R PENSPATH
0003.00      A            EMP_ID         9A
0004.00      A            PLAN_NAME     10A
0005.00      A            VESTED         1A
0006.00      A            AMT_CNTRB      6P 2
0007.00      A          K EMP_ID

VACATION Physical File DDS

0001.00      A                                      UNIQUE
0002.00      A          R VACPATH
0003.00      A            EMP_ID         9A
0004.00      A            DAYS_EARND     2P 1
0005.00      A            DAYS_TAKEN     2P 1
0006.00      A          K EMP_ID

EMP Multi-Format Logical DDS

0001.00      A          R EMPINFOR                   PFILE(EMPLOYEE)
0002.00      A          K EMP_ID
0003.00      A          R PENSPATH                   PFILE(PENSION)
0004.00      A          K EMP_ID
0005.00      A          R VACPATH                    PFILE(VACATION)
0006.00      A          K EMP_ID
```

When you create a synonym for the EMP Multi-format logical file, the following metadata is generated.

```
FILENAME=EMP_MFL, SUFFIX=DBFILE  ,
 DATASET=QSYS:SHIPPING/EMP, $
  SEGMENT=EMPINFOR, SEGTYPE=S0, $
    FIELDNAME=EMP_ID, ALIAS=EMP_ID, USAGE=A9, ACTUAL=A9,
      TITLE='EMP_ID', $
    FIELDNAME=LAST_NAME, ALIAS=LAST_NAME, USAGE=A10, ACTUAL=A10,
      TITLE='LAST_NAME', $
    FIELDNAME=FIRST_NAME, ALIAS=FIRST_NAME, USAGE=A10, ACTUAL=A10,
      TITLE='FIRST_NAME', $
    FIELDNAME=ADDRESS, ALIAS=ADDRESS, USAGE=A40, ACTUAL=A40,
      TITLE='ADDRESS', $
    FIELDNAME=RECFORM, ALIAS=EMPINFOR, USAGE=A10, ACTUAL=A10, $

  SEGMENT=PENSPATH, SEGTYPE=S0, PARENT=EMPINFOR, $
    FIELDNAME=EMP_ID, ALIAS=EMP_ID, USAGE=A9, ACTUAL=A9,
      TITLE='EMP_ID', $
    FIELDNAME=PLAN_NAME, ALIAS=PLAN_NAME, USAGE=A10, ACTUAL=A10,
      TITLE='PLAN_NAME', $
    FIELDNAME=VESTED, ALIAS=VESTED, USAGE=A1, ACTUAL=A1,
      TITLE='VESTED', $
    FIELDNAME=AMT_CNTRB, ALIAS=AMT_CNTRB, USAGE=P8.2, ACTUAL=P4,
      TITLE='AMT_CNTRB', $
    FIELDNAME=RECFORM, ALIAS=PENSPATH, USAGE=A10, ACTUAL=A10, $

  SEGMENT=VACPATH, SEGTYPE=S0, PARENT=PENSPATH, $
    FIELDNAME=EMP_ID, ALIAS=EMP_ID, USAGE=A9, ACTUAL=A9,
      TITLE='EMP_ID', $
    FIELDNAME=DAYS_EARND, ALIAS=DAYS_EARND, USAGE=P4.1, ACTUAL=P2,
      TITLE='DAYS_EARND', $
    FIELDNAME=DAYS_TAKEN, ALIAS=DAYS_TAKEN, USAGE=P4.1, ACTUAL=P2,
      TITLE='DAYS_TAKEN', $
    FIELDNAME=RECFORM, ALIAS=VACPATH, USAGE=A10, ACTUAL=A10, $
```

The internal field, RECFORM, is needed to reference a particular record format. (This is done automatically when you create the synonym.)

The RECFORM ALIAS is required, even for single format files, and must always have a value of the name of the record format.

DB Heritage Files Record Selection Efficiencies

The most efficient way to retrieve selected records from a data source is by applying an IF screening test against the primary key. This results in a direct reading of the data using the data source's index. Only those records that you request are retrieved from the file. The alternative method of retrieval, the sequential read, forces the adapter to retrieve all the records into storage.

Selection criteria that are based on the entire primary key, or on a subset of the primary key, cause direct reads using the index. A partial key is any contiguous part of the primary key beginning with the first byte.

IF selection tests performed against virtual fields can take advantage of these efficiencies as well, if the full or partial key is embedded in the virtual field.

The EQ and IS relations realize the greatest performance improvement over sequential reads. When testing on a partial key, equality logic is used to retrieve only the first segment instance of the screening value. To retrieve subsequent instances, NEXT logic is used.

Screening relations GE, FROM, FROM-TO, GT, EXCEEDS, IS-MORE-THAN, and NOT-FROM-TO all obtain some benefit from direct reads. The following example uses the index to find the record containing primary key value 66:

```
IF keyfield GE 66
```

It then continues to retrieve records by sequential processing, because DB Heritage Files stores records in ascending key sequence. The direct read is not attempted when the IF screening conditions NE, IS-NOT, CONTAINS, OMITS, LT, IS-LESS-THAN, LE, and NOT-FROM are used in the report request.

Reporting From Files With Alternate Indexes

Similar performance improvement is available for files that use alternate indexes. An alternate index provides access to records in a key sequenced data set based on a key other than the primary key.

All benefits and limitations inherent with screening on the primary or partial key are applicable to screening on the alternate index or partial alternate index.

Note: It is not necessary to take an explicit indexed view to use a particular index.

Chapter 21

Using the Adapter for Esri ArcGIS

The ESRI GIS Adapter is used to report against the information residing in the ESRI ArcGIS Online environment or on a locally hosted ArcGIS server. For example, Data Enrichment information for a specific ZIP code can be analyzed.

You can configure the ESRI GIS Adapter using the WebFOCUS Reporting Server Web Console. The adapter requires a connection, which stores the refresh token. A valid ESRI ArcGIS Online refresh token is required to issue ESRI ArcGIS Online API calls. This token is associated with an ESRI ArcGIS Online application and a specific ESRI ArcGIS Online user.

Note: To configure a named connection with the Adapter for Esri ArcGIS by clicking *Connect* on the WebFOCUS Home Page, the application server used by WebFOCUS and the WebFOCUS Reporting Server must both be running on the same machine. Otherwise, you cannot retrieve the refresh token for the adapter, and the Refresh Token field displays the text *Refresh Token unavailable*. In this case, configure the adapter from the Server Console, accessed through an HTTP listener, to acquire the token.

In this chapter:

Creating an ESRI ArcGIS Online Application

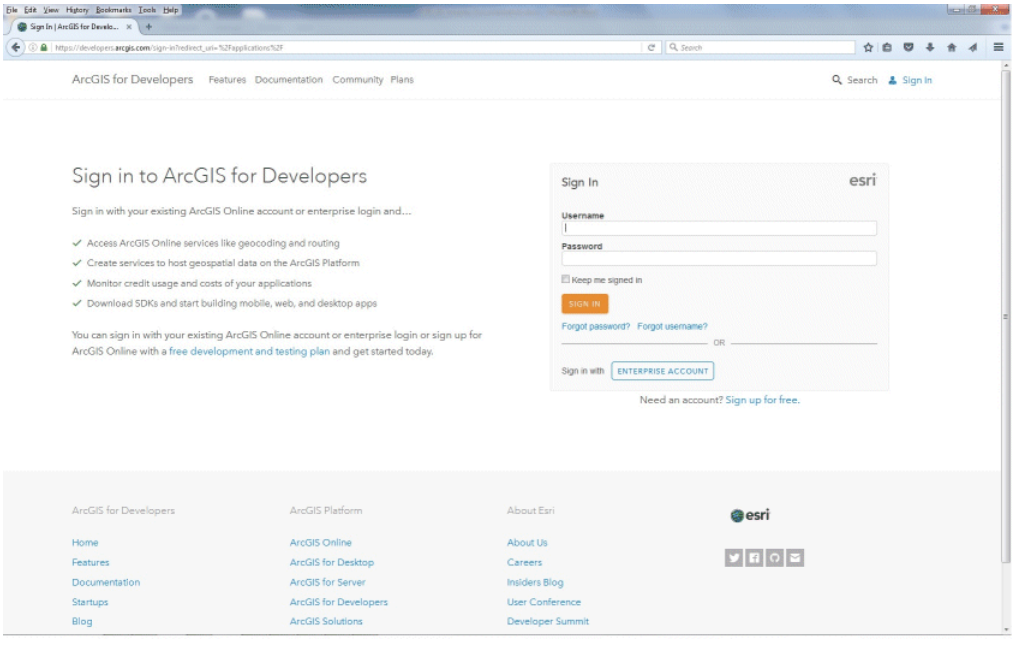
An ESRI ArcGIS Online application must be available before you can configure the ESRI GIS Adapter.

Procedure: How to Create an ESRI ArcGIS Application

1. Enter the following URL in a web browser:

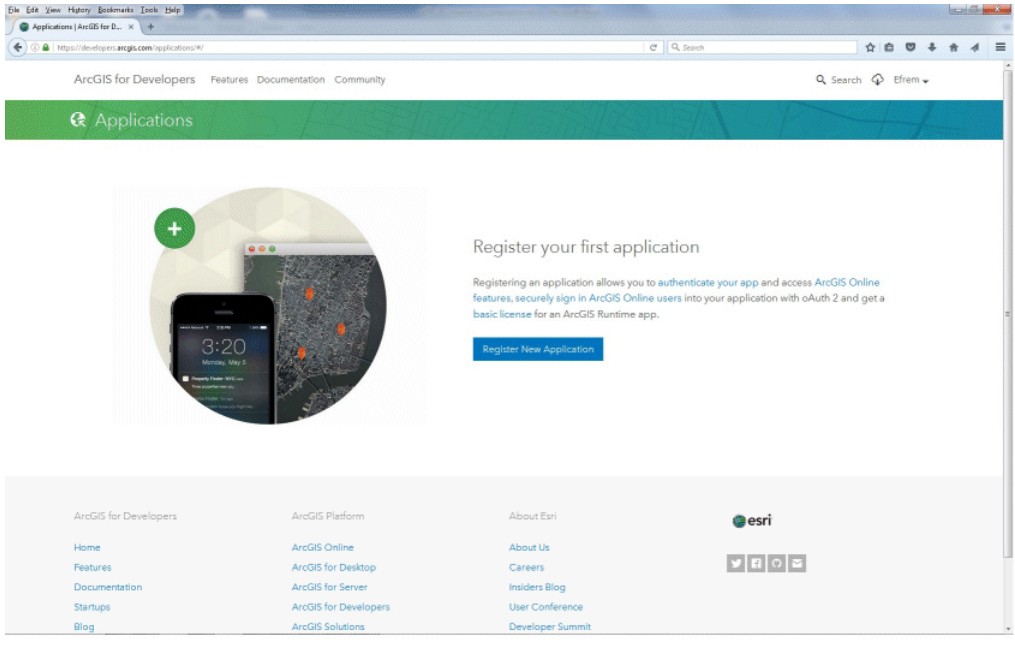
   `https://developers.arcgis.com/applications/#/`

If you are not already signed into the ESRI, a sign-in dialog for the ArcGIS for Developers site opens, as shown in the following image.



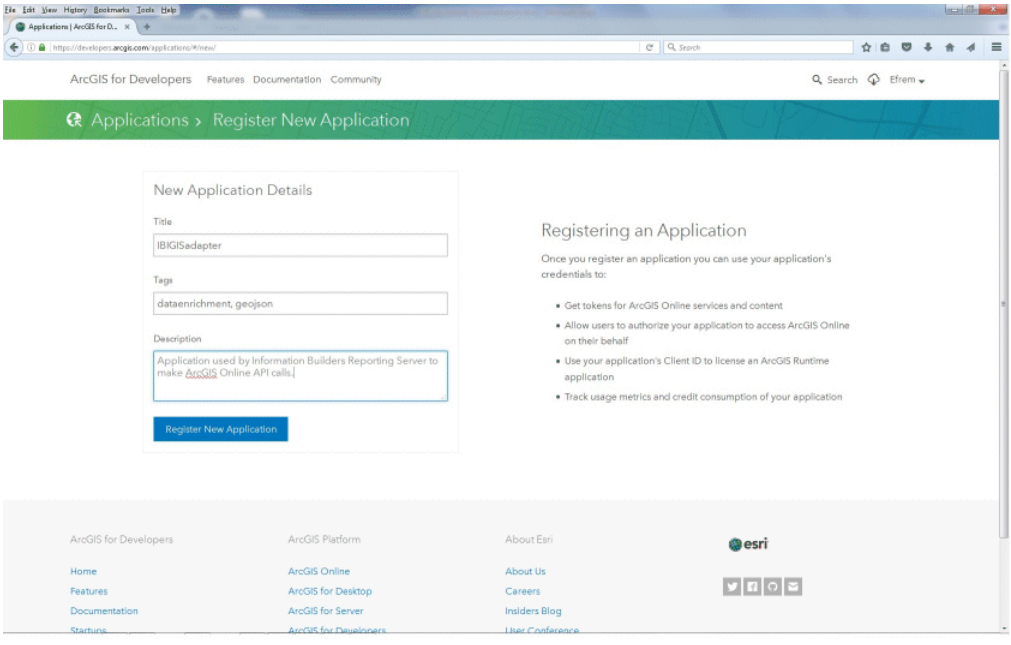
2. Enter valid ESRI ArcGIS Online account credentials that have the permission to create an ESRI ArcGIS Online application, and then click *Sign In*.

The Application screen opens, as shown in the following image.



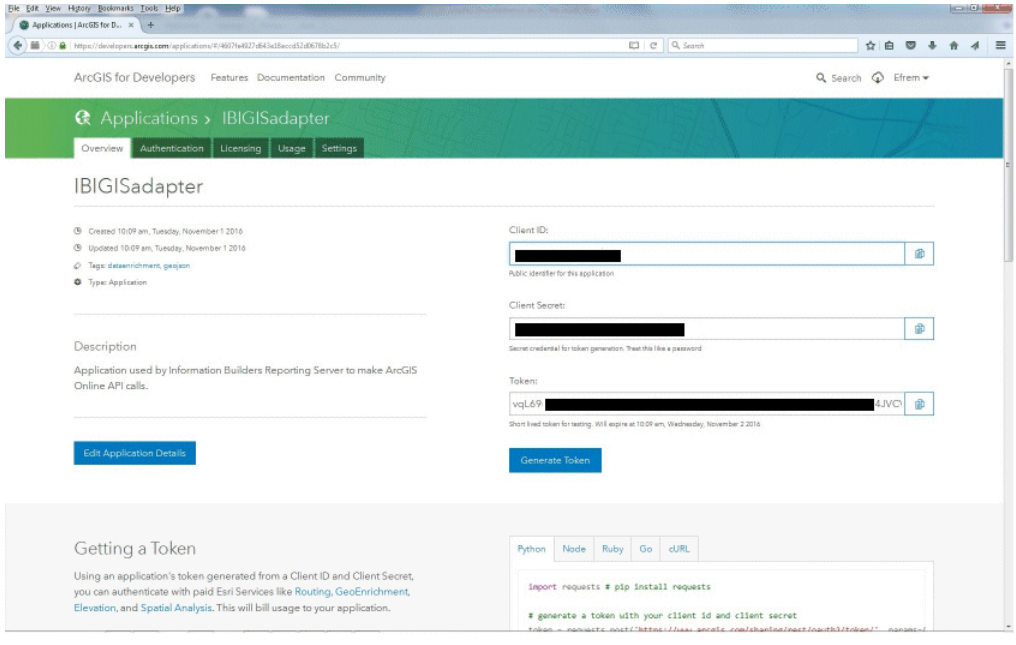
3. Click *Register New Application*.

The Register New Application screen opens, as shown in the following image.



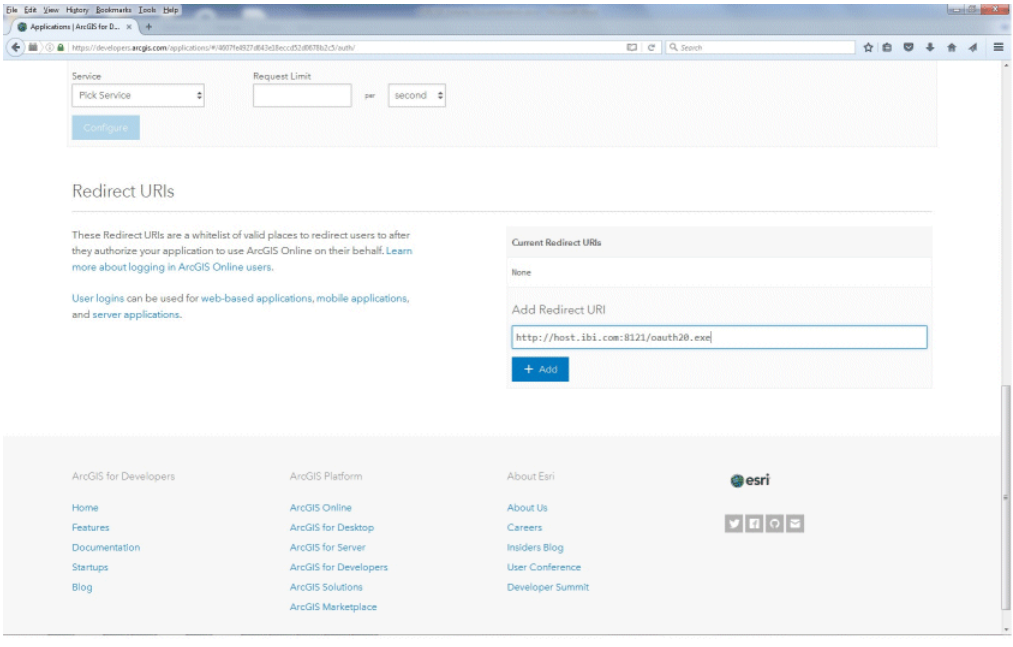
4. Enter a title, search tags, and a description for your new application and click *Register New Application*.

The Application Overview screen opens, which displays your Client ID and Client Secret values, as shown in the following image.



5. Click the *Authentication* tab.

Scroll down to the Redirect URIs section of the page, as shown in the following image.



6. Enter the host name and port used to access the Reporting Server Web Console appended with *oauth20.exe* in the Add Redirect URI field.

   For example:

   ```
   http://host.ibi.com:8121/oauth20.exe
   ```

   If the Reporting Server is installed as a standalone server, the following should be specified as the value in the Add Redirect URI field.

   ```
   http://localhost/oauth20.exe
   ```

7. Click + *Add*.

Configuring the Adapter for ESRI ArcGIS

This content describes how to configure the Adapter for ESRI ArcGIS.

Procedure: How to Configure the Adapter for ESRI ArcGIS for Non-Premium API Calls to ArcGIS Online

1. Clear cookies from the web browser that will be used to start the Reporting Server Web Console.

2. Access the Reporting Server Web Console using the host name and port that you specified in the Redirect URI field of the ESRI ArcGIS Online application.

   For example:

   ```
   http://host.ibi.com:8121
   ```

   For more information, see *Creating an ESRI ArcGIS Online Application* on page 599.

3. From the Web Console Applications page, click *Get Data*.

   or

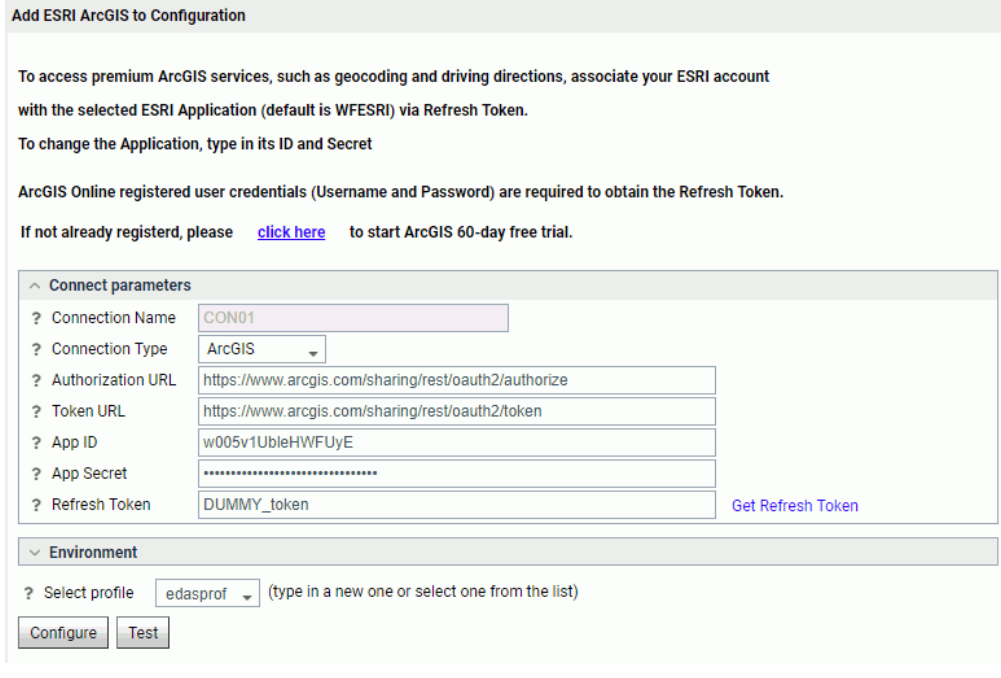
   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

4. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

5. Right-click the ESRI ArcGIS node and click *Add connection*.

   The Add ESRI ArcGIS to Configuration pane opens, as shown in the following image.

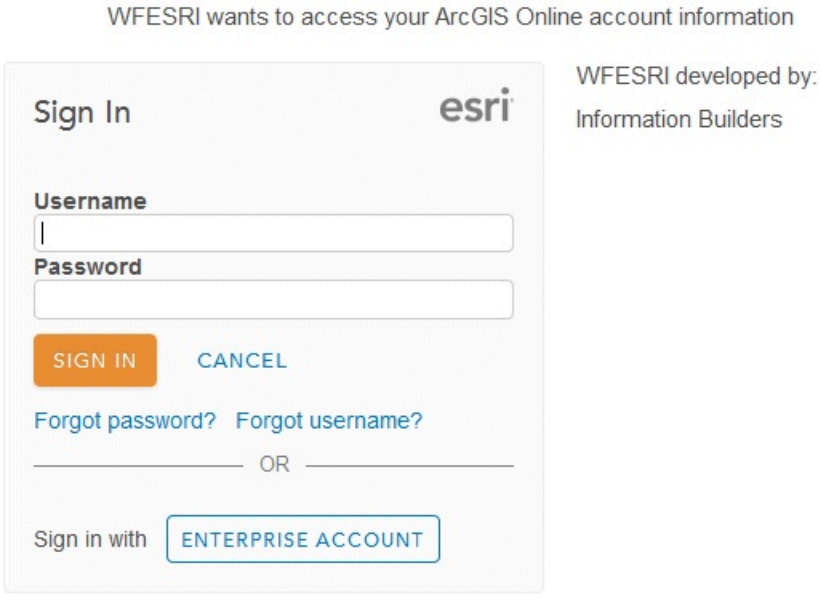


A default App ID and App Secret is prepopulated, to be used for the Non-Premium ESRI ArcGIS Online API calls covered in the Information Builders agreement.

Data Enrichment is included in the Premium ESRI ArcGIS Online API calls.

6. Click *Get Refresh Token*.

   The ESRI Sign In screen opens, as shown in the following image.



7. Enter the ESRI Sign In credentials and click *Sign In*.

   In order to obtain ESRI credentials, you can sign up at the following URL:

   https://developers.arcgis.com/sign-up/

   You are returned to the Add ESRI ArcGIS to Configuration pane where the Refresh Token is now populated.

8. Click *Configure*.

   The configured ESRI ArcGIS connection is added to the ESRI ArcGIS node in the left pane.

Procedure: How to Configure the Adapter for ESRI ArcGIS for Premium API Calls to ArcGIS Online

1. Clear cookies from the web browser that will be used to start the Reporting Server Web Console.

2. Access the Reporting Server Web Console using the host name and port that you specified in the Redirect URI field of the ESRI ArcGIS Online application.

   For example:

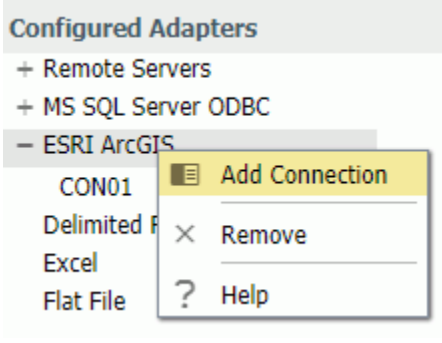
   ```
   http://host.ibi.com:8121
   ```

   For more information, see *Creating an ESRI ArcGIS Online Application* on page 599.

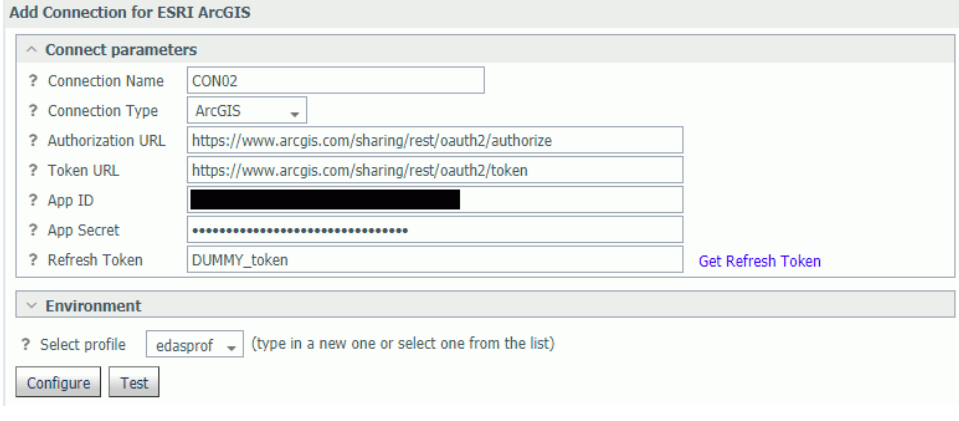
3. From the Reporting Server Web Console menu bar, click *Adapters*.

   The Adapters pane opens.

4. Expand the *Configured* folder, if it is not already expanded.
5. Expand the *GIS* folder.
6. Right-click the ESRI ArcGIS node and click *Add connection*, as shown in the following image.



   The Add Connection for ESRI ArcGIS pane opens, as shown in the following image.



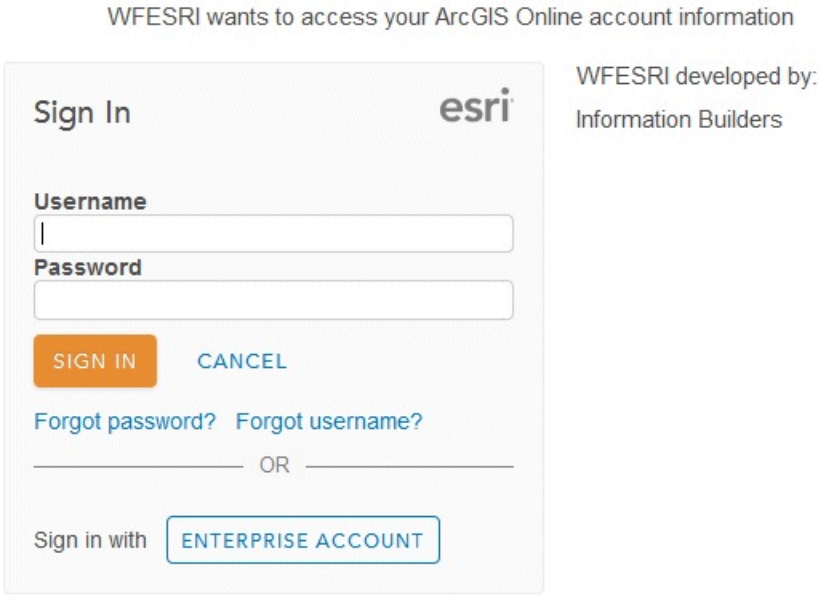
7. Enter the values for the App ID and App Secret defined in the ESRI application you created.

For more information, see *Creating an ESRI ArcGIS Online Application* on page 599.

Data Enrichment is included in the Premium ESRI ArcGIS Online API calls.

8. Click *Get Refresh Token*.

   The ESRI Sign In screen opens, as shown in the following image.



9. Enter the ESRI Sign In credentials and click *Sign In*.

   In order to obtain ESRI credentials, you can sign up at the following URL:

   https://developers.arcgis.com/sign-up/

   You are returned to the Add ESRI ArcGIS to Configuration pane where the Refresh Token is now populated.

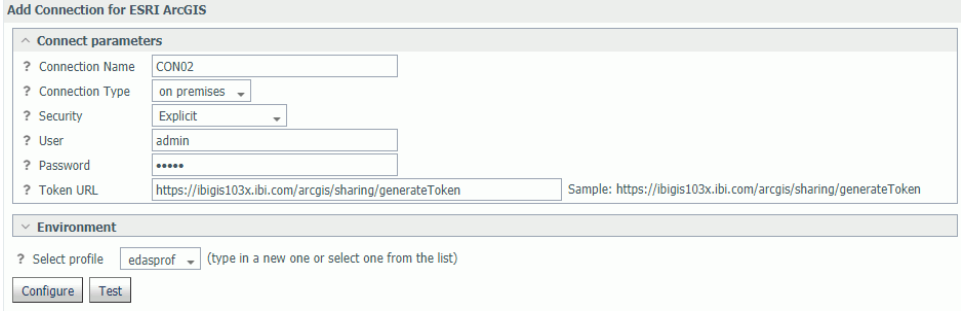
10. Click *Configure*.

    The configured ESRI ArcGIS connection is added to the ESRI ArcGIS node in the left pane.

Procedure: How to Configure a Locally Hosted ArcGIS Server

At some sites, ArcGIS Server is used in conjunction with or instead of ArcGIS Online. This type of configuration is called *Esri on Premise*.

1. When adding a connection for the Adapter for Esri ArcGIS, select *on premises* from the Connection Type drop-down list, as shown in the following image.



2. Select *Explicit* from the Security drop-down list.
3. Enter the user ID for the locally hosted ArcGIS Server in the User text box.
4. Enter the password for the locally hosted ArcGIS Server in the Password text box.
5. Enter the Token URL in the following format in the Token URL text box.

   `https://hostname/arcgis/sharing/generateToken`

   where:

   hostname

   Is the host where the ArcGIS Server is installed. For example:

   `https://ibigis103x.ibi.com/arcgis/sharing/generateToken`

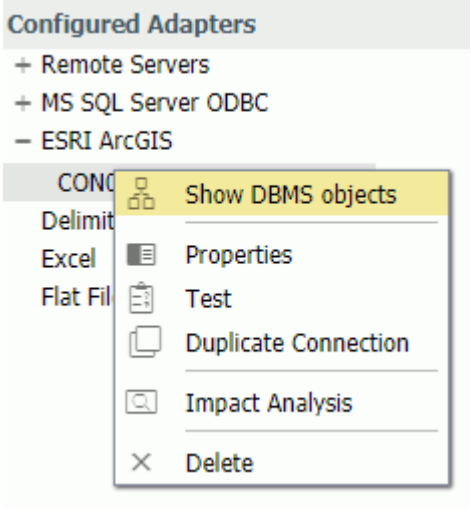
6. Test the connection by clicking *Test*.
7. When the test is successful, click *Configure*.

Creating Metadata and Sample Reports for the Adapter for ESRI ArcGIS Using Premium API Calls

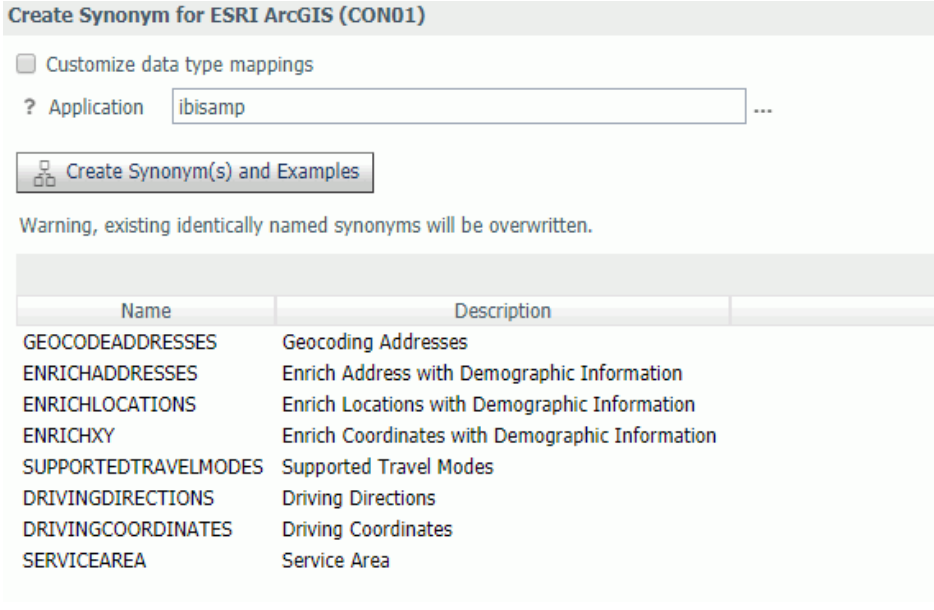
Create Synonym for the Adapter for ESRI ArcGIS creates the metadata and sample WebFOCUS reports used for WebFOCUS reporting against data returned from the Premium ESRI ArcGIS Online API calls.

Procedure: How to How to Create Metadata and Sample Reports for Premium API Calls

1. From the DMC, expand the *Adapters* folder, then the *Configured* folder, and then the *ESRI ArcGIS* folder.
2. Right-click the configured connection for the ESRI ArcGIS connection (for example, CON02) that has access to Premium ArcGIS Online API calls, and click *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for ESRI ArcGIS pane opens, as shown in the following image.



3. Enter a specific application in the Application field, or click the ellipsis button to the right of the field to select an application where the metadata and sample reports are to be stored.
4. Click *Create Synonym(s) and Example*s.

   The Create Synonym for ESRI ArcGIS Status pane opens and indicates that the synonym was created successfully.

Sample Metadata and Reports

This section describes the metadata and sample reports for the Adapter for ESRI ArcGIS.

Reference: Adapter for ESRI ArcGIS Metadata

Metadata	Description
drivingdirections	Driving directions between geocoded coordinates
enrichaddresses	Enriches a radius around an address with demographic information

Metadata	Description
enrichlocations	Enriches locations with demographic information (for example, ZIP codes)
enrichxy	Enriches a radius around a geocoded coordinate with demographic information
geocodeaddresses	Geocodes specific addresses
servicearea	Geocoded coordinates that can be serviced around a specific location
supportedtravelmodes	Valid travel modes

Reference: Adapter for ESRI ArcGIS Sample Reports

The following table lists and describes the sample reports for the Adapter for ESRI ArcGIS.

Sample Report	Description
esrsampl/drivingcoordinates	Returns a series of geocoded coordinates for the driving directions between geocoded coordinates. Uses drivingdirections synonym.
esrsampl/drivingdirections	Returns the driving directions between geocoded coordinates. Uses drivingdirections synonym.
esrsampl/enrich_wfretail_us	Ennriches a ZIP code from the Retail Sales Demo Data with demographic information. Uses enrichlocations synonym.
esrsampl/enrichaddresses	Enriches a radius around an address with demographic information. Uses enrichaddresses synonym.

Sample Report	Description
esrsampl/enrichlocations	Enriches one or more ZIP codes with demographic information. Uses enrichlocations synonym.
esrsampl/enrichxy	Enriches a radius around a geocoded coordinate with demographic information. Uses enrichxy synonym.
esrsampl/geocodeaddresses	Geocodes specific addresses. Uses geocodeaddresses synonym.
esrsampl/servicearea	Returns a list of geocoded coordinates that can be serviced around a specific location. Uses servicearea synonym.
esrsampl/supportedtravelmodes	Returns a list of supported travel modes. Uses supportedtravelmodes synonym.

Chapter 22

Using the Adapter for Essbase

The Adapter for Essbase allows applications to access Essbase data sources. The adapter converts application requests into native Essbase statements and returns optimized answer sets to the requesting application.

The Adapter for Essbase provides access to Hyperion Cubes from a server running on Microsoft Windows or UNIX. Essbase can only be accessed using an API supplied by the vendor. It is not possible to gain access using Direct SQL (Direct Passthru in server terminology) or the Adapter for ODBC.

Note: The level of User and Group Permissions for Applications/Databases required by Essbase 11.1.2 in order to be able to read the Database Outline was changed to the Access Database/Database Manager Level. All Reporting Server requests against Essbase read the Database Outline and, therefore, require this level of permissions.

In this chapter:

Preparing the Essbase Environment

The Adapter for Essbase minimally requires the installation of the Essbase Client. The Essbase Client allows you to connect to a local or remote Essbase Analytical Server.

Note: The level of User and Group Permissions for Applications/Databases required by Essbase 11.1.2 in order to be able to read the Database Outline was changed to the Access Database/Database Manager Level. All Reporting Server requests against Essbase read the Database Outline and, therefore, require this level of permissions.

Procedure: How to Prepare the Environment on Microsoft Windows

On Microsoft Windows, the Essbase environment is set up during the installation of Essbase.

Procedure: How to Prepare the Environment on UNIX

The following variables must be set during the installation of the Essbase Client:

ARBORPATH

Is the home directory for the Essbase Client.

For example, for the Essbase Client 11.1.2 on UNIX:

```
ARBORPATH= /rdbms5/arb11cli/EPMSystem11R1/products/Essbase/EssbaseClient
export ARBORPATH
```

ESSBASEPATH

Is the home directory for the Essbase Client (same as ARBORPATH).

For example, for the Essbase Client 11.1.2 on UNIX:

```
ESSBASEPATH= /rdbms5/arb11cli/EPMSystem11R1/products/Essbase/
EssbaseClient
export ESSBASEPATH
```

ESSLANG

Is the language setting for the Essbase Client.

For example, for the Essbase Client 11.1.2 on UNIX:

```
ESSLANG=English_UnitedStates.US-ASCII@Binary
export ESSLANG
```

LD_LIBRARY_PATH

Is the path for the Essbase Client libraries. LD_LIBRARY_PATH must include the bin directory under the Essbase Client home directory as well as the Essbase Client home directory, in that order.

For example, for the Essbase Client 11.1.2 on UNIX:

```
LD_LIBRARY_PATH =
/rdbms5/arb11cli/EPMSystem11R1/products/Essbase/EssbaseClient/bin;
/rdbms5/arb11cli/EPMSystem11R1/products/Essbase/EssbaseClient
export LD_LIBRARY_PATH
```

Configuring the Adapter for Essbase

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

When connecting to an Essbase Analytical Server, you must use Explicit authentication or Password passthru. This method requires that user IDs and passwords be explicitly stated in the SET CONNECTION_ATTRIBUTES command. You can include these commands in the server global profile, edasprof.prf, for all users. You can also issue the SET DEFAULT_CONNECTION command in the edasprof.prf file to select one of these connections as the default connection to use if the Access File does not include a server or connection name.

When you access an Essbase Analytical Server using the Reporting Server, you are identified by a user ID and password. The Database Administrator can assign management privileges to a specific user, where privileges:

- Are constant for all applications and databases.
- Apply to an application.
- Apply to a database.

For more information on security, see the *Hyperion Essbase* documentation.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Essbase

The *Essbase* adapter in under the *OLAP* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server

Machine name where Essbase is running.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

The Password Passthru option requires that the server be started with DBMS security for Essbase Server authentication. See the *Server Administration* manual for information about configuring DBMS security.

User

User name by which you are known to the database.

Password

Password associated with the user name.

Outline Access

Existing connection name which is assumed to have Essbase Database Outline access and will be referenced by users having no Database Outline access privileges.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax:

How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Essbase, at connection time, for authentication.

```
ENGINE ESSBASE SET CONNECTION_ATTRIBUTES [connection]server
user/pwd
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Essbase, at connection time, for authentication. This option requires that the server be started with security set to DBMS.

```
ENGINE ESSBASE SET CONNECTION_ATTRIBUTES [connection]server
```

where:

ESSBASE

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the server name used as a connect descriptor to the Essbase Server across the network.

server

Is the machine where Essbase is running.

userid

Is the primary authorization ID by which you are known to Essbase.

password

Is the password associated with the primary authorization ID.

Testing an Essbase Connection

Once connections have been defined in the edasprof.prf server global profile, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection, which is used if the Access File does not specify connection name or server. You can override this default using the SET DEFAULT_CONNECTION command.

Procedure: How to Test an Essbase Connection

There are two ways to access the Test function:

Right-click a connection and select *Test* from the pop-up menu.

or

1. Right-click a connection and select *Properties* from the pop-up menu.

   The Change Connect Parameters pane opens.

2. Click the *Test* button.

Managing Essbase Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Essbase data type.

Creating Synonyms

Synonyms define unique names for each Essbase *application.database* combination that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Essbase

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select a database

Select one or more databases from the list.

Make alias fields

Select the *Make alias fields* check box if you wish to create fields that expose the values in your alias table. The alias table can provide meaningful field values or field values in a specific language.

Aggregate

This option enables you to control summing on non-aggregated fields in an Essbase request.

- Choose *On* to enable summing on non-aggregated fields.
- Choose *Off* to disable summing on non-aggregated fields. For additional information and syntax, see *Preventing Aggregation of Non-Consolidating Members* on page 644.

Select Measure

This option enables you to limit the number of fields in the Master File.

- Select *none* to create a single data value representing the cube measures.
- Select a dimension to create a separate numeric measure field for each member in that dimension.

  Note: If you select a dimension, you can select an additional dimension for grouping the measures.

Select Measure Group

If you selected a dimension to expose the measures, you can optionally select an additional dimension for grouping those measures.

Note: You can then select specific members to create as groups.

Measure Group members

If you selected a Measure Group, you can now select specific members for which to create groups.

1. Click the *Select Measure Group members* check box.
2. Click *Next*.

   A list of Scenario members displays.
3. Check the members that you want to create as groups. Check the *Scenario Name* check box to create groups for all members.

Synonym field name processing options:

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Synonym Name

To change the default name, type the name of the synonym.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Example: Creating a Synonym

To generate a synonym for the application *Sample* and database *Basic*, enter the following information on the Create Synonym panes of the Web Console or the Data Management Console:

1. From the displayed list, select the row that contains the Application Name *Sample* and the Data Base Name *Basic*, then click the *Next* button.
2. Accept the default for *Measure*.
3. Click *Create Synonym*. The synonym is created and added under the specified application directory (baseapp is the default).

   A status window displays a message indicating that the synonym was created successfully.
4. Open the *baseapp* application folder in the navigation pane and click the synonym *Basic*.
5. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File

```
FILENAME=SAMPLE, SUFFIX=ESSBASE, $
  SEGMENT=BASIC, SEGTYPE=S0, $
$  DIMENSION: Year
 DIMENSION=Year, CAPTION='Year', $
  HIERARCHY=Year, CAPTION='Year Levels', HRY_DIMENSION=Year,
     HRY_STRUCTURE=STANDARD, $
    FIELDNAME=YEAR, ALIAS=History, USAGE=A4, ACTUAL=A4,
      WITHIN='*Year', PROPERTY=UID,  $
    FIELDNAME=QUARTER, ALIAS=Quarter, USAGE=A8, ACTUAL=A8,
      WITHIN=YEAR, PROPERTY=UID,  $
    FIELDNAME=MONTH, ALIAS=Month, USAGE=A9, ACTUAL=A9,
      WITHIN=QUARTER, PROPERTY=UID,  $
```

```
  HIERARCHY=Year2, CAPTION='Year Parent-Child', HRY_DIMENSION=Year,
      HRY_STRUCTURE=RECURSIVE, $
    FIELDNAME=YEAR_MEMBER, ALIAS=Year, USAGE=A4, ACTUAL=A4,
      WITHIN='*Year2', PROPERTY=UID,  $
    FIELDNAME=YEAR_CAPTION, USAGE=A9, ACTUAL=A9,
      REFERENCE=YEAR_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=YEAR_PARENT, USAGE=A4, ACTUAL=A4,
      REFERENCE=YEAR_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=YEAR_PARENTCAP, USAGE=A9, ACTUAL=A9,
      REFERENCE=YEAR_MEMBER, PROPERTY=CAP_PARENT,  $
    FIELDNAME=YEAR_LVLNO, USAGE=I2L, ACTUAL=I4,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      TITLE='Year_LVLNO LEVEL_NUMBER',
      REFERENCE=YEAR_MEMBER, PROPERTY=LEVEL_NUMBER,  $
    FIELDNAME=H_T_D, ALIAS='H-T-D', USAGE=A16, ACTUAL=A16,
      REFERENCE=YEAR, PROPERTY=TIMESERIES,  $
    FIELDNAME=Q_T_D, ALIAS='Q-T-D', USAGE=A16, ACTUAL=A16,
      REFERENCE=QUARTER, PROPERTY=TIMESERIES,  $
    FIELDNAME=M_T_D, ALIAS='M-T-D', USAGE=A16, ACTUAL=A16,
      REFERENCE=MONTH, PROPERTY=TIMESERIES,  $

$  DIMENSION: Measures
 DIMENSION=Measures, CAPTION='Measures', $
  HIERARCHY=Measures, CAPTION='Measures Levels', HRY_DIMENSION=Measures,
   HRY_STRUCTURE=STANDARD, $
    FIELDNAME=MEASURES, ALIAS='Gen1,Measures', USAGE=A8, ACTUAL=A8,
      WITHIN='*Measures', PROPERTY=UID,  $
    FIELDNAME=GEN2_MEASURES, ALIAS='Gen2,Measures', USAGE=A9, ACTUAL=A9,
      WITHIN=MEASURES, PROPERTY=UID,  $
    FIELDNAME=GEN3_MEASURES, ALIAS='Gen3,Measures', USAGE=A17,
      ACTUAL=A17, WITHIN=GEN2_MEASURES, PROPERTY=UID,  $
    FIELDNAME=GEN4_MEASURES, ALIAS='Gen4,Measures', USAGE=A18,
      ACTUAL=A18, WITHIN=GEN3_MEASURES, PROPERTY=UID,  $

  HIERARCHY=Measures2, CAPTION='Measures Parent-Child',
   HRY_DIMENSION=Measures, HRY_STRUCTURE=RECURSIVE, $
    FIELDNAME=MEASURES_MEMBER, ALIAS=Measures, USAGE=A17, ACTUAL=A17,
      WITHIN='*Measures2', PROPERTY=UID,  $
    FIELDNAME=MEASURES_CAPTION, USAGE=A18, ACTUAL=A18,
      REFERENCE=MEASURES_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=MEASURES_PARENT, USAGE=A17, ACTUAL=A17,
      REFERENCE=MEASURES_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=MEASURES_PARENTCAP, USAGE=A18, ACTUAL=A18,
      REFERENCE=MEASURES_MEMBER, PROPERTY=CAP_PARENT,  $
    FIELDNAME=MEASURES_LVLNO, USAGE=I2L, ACTUAL=I4,MISSING=ON,
      ACCESS_PROPERTY=(INTERNAL),TITLE='Measures_LVLNO LEVEL_NUMBER',
      REFERENCE=MEASURES_MEMBER, PROPERTY=LEVEL_NUMBER,  $
```

```
$  DIMENSION: Product
 DIMENSION=Product, CAPTION='Product', $
  HIERARCHY=Product, CAPTION='Product Levels', HRY_DIMENSION=Product,
   HRY_STRUCTURE=STANDARD, $
    FIELDNAME=PRODUCT, ALIAS='Lev2,Product', USAGE=A7, ACTUAL=A7,
      WITHIN='*Product',PROPERTY=UID,  $
    FIELDNAME=FAMILY, ALIAS=Family, USAGE=A11, ACTUAL=A11,
      WITHIN=PRODUCT,PROPERTY=UID,  $
    FIELDNAME=SKU, ALIAS=SKU, USAGE=A18, ACTUAL=A18,
      WITHIN=FAMILY,PROPERTY=UID,  $

  HIERARCHY=Product2, CAPTION='Product Parent-Child',
   HRY_DIMENSION=Product, HRY_STRUCTURE=RECURSIVE, $
    FIELDNAME=PRODUCT_MEMBER, ALIAS=Product, USAGE=A7, ACTUAL=A7,
      WITHIN='*Product2',PROPERTY=UID,  $
    FIELDNAME=PRODUCT_CAPTION, USAGE=A18, ACTUAL=A18,
      REFERENCE=PRODUCT_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=PRODUCT_PARENT, USAGE=A7, ACTUAL=A7,
      REFERENCE=PRODUCT_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=PRODUCT_PARENTCAP, USAGE=A18, ACTUAL=A18,
      REFERENCE=PRODUCT_MEMBER, PROPERTY=CAP_PARENT,  $
    FIELDNAME=PRODUCT_LVLNO, USAGE=I2L, ACTUAL=I4,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      TITLE='Product_LVLNO LEVEL_NUMBER',
      REFERENCE=PRODUCT_MEMBER, PROPERTY=LEVEL_NUMBER,  $

$  DIMENSION: Market
 DIMENSION=Market, CAPTION='Market', $
  HIERARCHY=Market, CAPTION='Market Levels', HRY_DIMENSION=Market,
   HRY_STRUCTURE=STANDARD, $
    FIELDNAME=MARKET, ALIAS='Gen1,Market', USAGE=A6, ACTUAL=A6,
      WITHIN='*Market', PROPERTY=UID,  $
    FIELDNAME=REGION, ALIAS=Region, USAGE=A7, ACTUAL=A7,
      WITHIN=MARKET, PROPERTY=UID,  $
    FIELDNAME=STATE, ALIAS=State, USAGE=A13, ACTUAL=A13,
      WITHIN=REGION, PROPERTY=UID,  $

  HIERARCHY=Market2, CAPTION='Market Parent-Child', HRY_DIMENSION=Market,
   HRY_STRUCTURE=RECURSIVE, $
    FIELDNAME=MARKET_MEMBER, ALIAS=Market, USAGE=A13, ACTUAL=A13,
      WITHIN='*Market2', PROPERTY=UID,  $
    FIELDNAME=MARKET_CAPTION, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=MARKET_PARENT, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=MARKET_PARENTCAP, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=CAP_PARENT,  $
    FIELDNAME=MARKET_LVLNO, USAGE=I2L, ACTUAL=I4, MISSING=ON,
      ACCESS_PROPERTY=(INTERNAL), TITLE='Market_LVLNO LEVEL_NUMBER',
      REFERENCE=MARKET_MEMBER, PROPERTY=LEVEL_NUMBER,  $
```

```
$  DIMENSION: Scenario
 DIMENSION=Scenario, CAPTION='Scenario', $
  HIERARCHY=Scenario, CAPTION='Scenario Levels', HRY_DIMENSION=Scenario,
   HRY_STRUCTURE=STANDARD, $
    FIELDNAME=SCENARIO, ALIAS='Gen1,Scenario', USAGE=A8, ACTUAL=A8,
      WITHIN='*Scenario', PROPERTY=UID,  $
    FIELDNAME=GEN2_SCENARIO, ALIAS='Gen2,Scenario', USAGE=A10,
      ACTUAL=A10, WITHIN=SCENARIO, PROPERTY=UID,  $

  HIERARCHY=Scenario2, CAPTION='Scenario Parent-Child',
   HRY_DIMENSION=Scenario, HRY_STRUCTURE=RECURSIVE, $
    FIELDNAME=SCENARIO_MEMBER, ALIAS=Scenario, USAGE=A10, ACTUAL=A10,
      WITHIN='*Scenario2', PROPERTY=UID,  $
    FIELDNAME=SCENARIO_CAPTION, USAGE=A10, ACTUAL=A10,
      REFERENCE=SCENARIO_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=SCENARIO_PARENT, USAGE=A10, ACTUAL=A10,
      REFERENCE=SCENARIO_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=SCENARIO_PARENTCAP, USAGE=A10, ACTUAL=A10,
      REFERENCE=SCENARIO_MEMBER, PROPERTY=CAP_PARENT,  $
    FIELDNAME=SCENARIO_LVLNO, USAGE=I2L, ACTUAL=I4,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      TITLE='Scenario_LVLNO LEVEL_NUMBER',
      REFERENCE=SCENARIO_MEMBER, PROPERTY=LEVEL_NUMBER,  $

$  DIMENSION: DATA
  SEGMENT=DATA, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=DATA_VALUE, ALIAS=DATA_VALUE, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, TITLE='DATA_VALUE', $

$  DIMENSION: ATTR
  SEGMENT=ATTR, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=CAFFEINATED, ALIAS=Caffeinated, USAGE=A17, ACTUAL=A17,
      REFERENCE=SKU, PROPERTY=UDA,  $
    FIELDNAME=OUNCES, ALIAS=Ounces, USAGE=A9, ACTUAL=A9,
      REFERENCE=SKU, PROPERTY=UDA,  $
    FIELDNAME=PKG_TYPE, ALIAS='Pkg Type', USAGE=A8, ACTUAL=A8,
      REFERENCE=SKU, PROPERTY=UDA,  $
    FIELDNAME=POPULATION, ALIAS=Population, USAGE=A15, ACTUAL=A15,
      REFERENCE=STATE, PROPERTY=UDA,  $
    FIELDNAME=INTRO_DATE, ALIAS='Intro Date', USAGE=A21, ACTUAL=A21,
      REFERENCE=SKU, PROPERTY=UDA,  $

$  DIMENSION: UDA
  SEGMENT=UDA, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=MARKET_UDA, ALIAS=Market, USAGE=A13, ACTUAL=A13,
      REFERENCE=**Market, PROPERTY=UDA,  $
```

Generated Access File

```
SEGNAME=BASIC, SERVER=EDASOL28, DBNAME=Basic, APPLNAME=Sample, $
 TIMEDIM=YEAR, $
    SHARE=300-30, PARENT=Diet, DIM=Product, $
    SHARE=200-20, PARENT=Diet, DIM=Product, $
    SHARE=100-20, PARENT=Diet, DIM=Product, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Using Default Field Names

The default field names that are used are taken from the generation names, if available, in the outline. Otherwise, they take the format

FIELDNAME=*generationnumber_dimensionname*

where:

generationnumber

Is the generation number within the dimension (for example, GEN2).

dimensionname

Is the name of the dimension.

For example, if the outline has a SCENARIO dimension, the first field name would be FIELDNAME=Scenario to represent the highest generation, the second field name would be FIELDNAME=GEN2_SCENARIO, and so on down the dimension hierarchy. For an illustration, see *Creating a Synonym* on page 624.

Reference: Access File Keywords

Keyword	Description
SERVER	Essbase Server name.
DBNAME=*database_name*	Indicates access to the specified database within an application.

Keyword	Description
APPLNAME=*application_name*	Indicates access to the specified application, which can contain one or more databases.

Parent/Child Support

The parent/child view in the Master File enables you to make requests using a member without having to know the generation to which the member belongs. The fields that provide this functionality are:

DIMENSION_MEMBER

Is the declaration for the field that contains the dimension members. It represents members of product at all levels of the dimension.

DIMENSION_CAPTION

Is the declaration for the field that contains the label displayed on reports for DIMENSION_MEMBER (this is the Essbase alias).

DIMENSION_PARENT

Is the declaration for the field that contains the parent of DIMENSION_MEMBER.

DIMENSION_PARENTCAP

Is the declaration for the field that contains the label displayed on reports for DIMENSION_PARENT (this is the Essbase alias).

The following is a sample of the parent/child view in the Master File:

```
FIELDNAME=SCENARIO_MEMBER, ALIAS=Scenario, USAGE=A10, ACTUAL=A10,
      WITHIN='*Scenario2',
      PROPERTY=UID,  $
    FIELDNAME=SCENARIO_CAPTION, USAGE=A10, ACTUAL=A10,
      REFERENCE=SCENARIO_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=SCENARIO_PARENT, USAGE=A10, ACTUAL=A10,
      REFERENCE=SCENARIO_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=SCENARIO_PARENTCAP, USAGE=A10, ACTUAL=A10,
      REFERENCE=SCENARIO_MEMBER, PROPERTY=CAP_PARENT,  $
```

Example: Using the Parent/Child View in the Master File

In the first request, Actual is explicitly identified with the generation GEN2_SCENARIO:

```
TABLE FILE BASIC
PRINT DATA_VALUE
WHERE GEN2_SCENARIO EQ 'Actual'
END
```

In the second request, which takes advantage of a parent/child view, Actual is represented as a member of the Scenario dimension. It is not necessary to know which generation it falls under.

```
TABLE FILE BASIC
PRINT DATA_VALUE
WHERE SCENARIO_MEMBER EQ 'Actual'
END
```

Both requests yield the same output:

```
DATA_VALUE = 105,522.00
```

Support for User-Defined Attributes

A User-Defined Attribute (UDA) in Essbase enables you to select and report on data based on a common characteristic. You can include UDAs in report requests. For each dimension with UDAs in an Essbase outline, synonym creation generates a UDA field name under a segment called UDA in the Master File.

Example: Reporting From a UDA Segment

The Sample Master File contains the following UDA segment:

```
$ DIMENSION: UDA
  SEGMENT=UDA, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=MARKET_UDA, ALIAS=Market, USAGE=A13, ACTUAL=A13,
      REFERENCE=**Market, PROPERTY=UDA,  $
```

When referencing UDAs in a request, you must also reference a member of the dimension that contains the UDA. In this example, STATE is a member of the MARKET Dimension in the Master File.

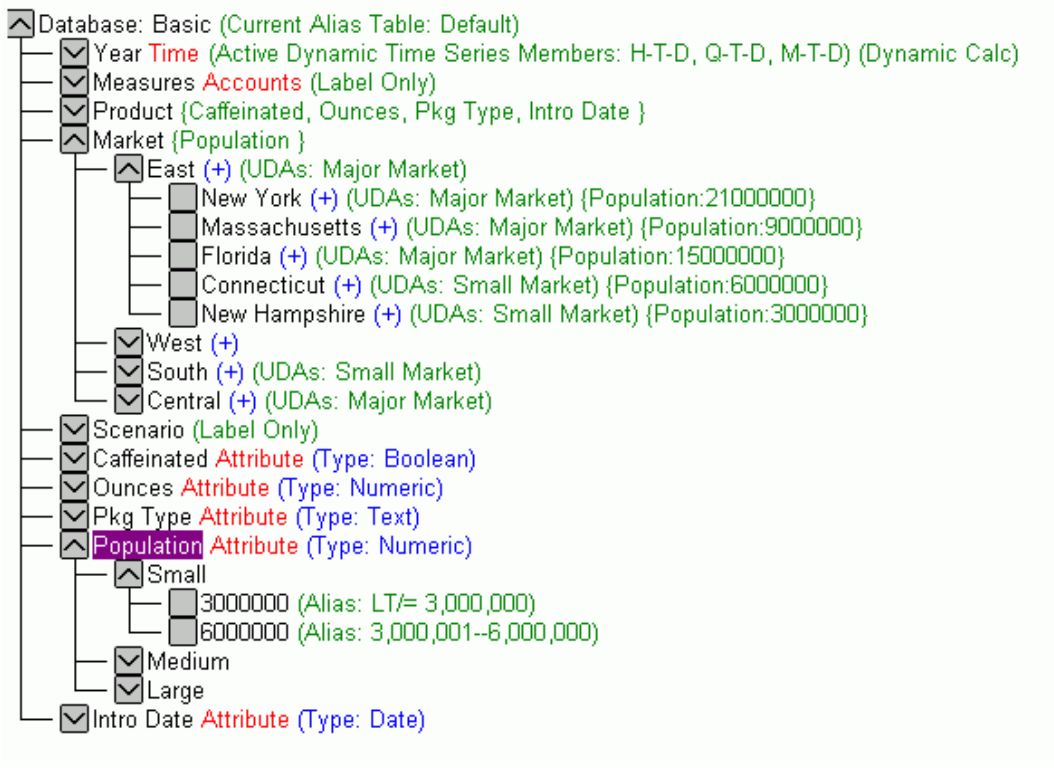
```
TABLE FILE BASIC
PRINT DATA_VALUE BY STATE
WHERE MARKET_UDA EQ 'New Market'
END
```

The output displays the values only for the states that have the UDA 'New Market' as a common characteristic:

```
STATE      DATA_VALUE
---------  ----------
Colorado     7,227.00
Louisiana    2,992.00
Nevada       4,039.00
```

Describing Attribute Dimensions in the Master File

An attribute dimension is identified by the word Attribute, which appears next to the Dimension name in the Essbase outline. Attribute dimensions are usually associated with standard Essbase dimensions. For example, in the following outline, Population is an Attribute dimension associated with the standard dimension Market.



The standard dimension serves as the base dimension for the associated attribute dimensions.

Example: Generating Attribute Dimensions

Synonym creation generates a Master File that contains an ATTR (attribute) segment, which defines attribute tagged dimensions.

The following is a portion of the Master File BASIC. Notice that Market is the base dimension with which the attribute dimension Population is associated.

```
FILENAME=SAMPLE, SUFFIX=ESSBASE ,$
  SEGMENT=BASIC, SEGTYPE=S0, $
>.
>.
>.
$  DIMENSION: Market
 DIMENSION=Market,CAPTION=Market, $
  HIERARCHY=Market,CAPTION=Market Levels,
     HRY_DIMENSION=Market,HRY_STRUCTURE=S, $
    FIELDNAME=MARKET, ALIAS='Gen1,Market', USAGE=A6, ACTUAL=A6,
      WITHIN='*Market', PROPERTY=UID,   $
    FIELDNAME=REGION, ALIAS=Region, USAGE=A7, ACTUAL=A7,
      WITHIN=MARKET,PROPERTY=UID, $
    FIELDNAME=STATE, ALIAS=State, USAGE=A13, ACTUAL=A13,
      WITHIN=REGION,PROPERTY=UID, $
  HIERARCHY=Market2,CAPTION=Market Parent-Child,
      HRY_DIMENSION=Market,HRY_STRUCTURE=R, $
    FIELDNAME=MARKET_MEMBER, ALIAS=Market, USAGE=A13, ACTUAL=A13,
      WITHIN='*Market2', PROPERTY=UID,  $
    FIELDNAME=MARKET_CAPTION, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=CAPTION,  $
    FIELDNAME=MARKET_PARENT, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=PARENT_OF,  $
    FIELDNAME=MARKET_PARENTCAP, USAGE=A13, ACTUAL=A13,
      REFERENCE=MARKET_MEMBER, PROPERTY=CAP_PARENT, $
.
.
.
  DIMENSION: ATTR
  SEGMENT=ATTR, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=CAFFEINATED, ALIAS=Product, USAGE=A17, ACTUAL=A17,
      REFERENCE=SKU, PROPERTY=UDA,  $
    FIELDNAME=OUNCES, ALIAS=Product, USAGE=A9, ACTUAL=A9,
      REFERENCE=SKU, PROPERTY=UDA,  $
    FIELDNAME=PKG_TYPE, ALIAS=Product, USAGE=A8, ACTUAL=A8,
      REFERENCE=SKU, PROPERTY=UDA,  $
    FIELDNAME=POPULATION, ALIAS=Market, USAGE=A15, ACTUAL=A15,
      REFERENCE=STATE, PROPERTY=UDA,  $
    FIELDNAME=INTRO_DATE, ALIAS=Product, USAGE=A21, ACTUAL=A21,
      REFERENCE=SKU, PROPERTY=UDA,  $
```

The corresponding Access File contains any two-pass calculated members, as well as members with the following consolidation properties: (-0, (/), (*), (%). In addition, it contains the names of any shared dimension members. For an illustration, see *Reporting Against Attribute Tagged Dimensions* on page 633.

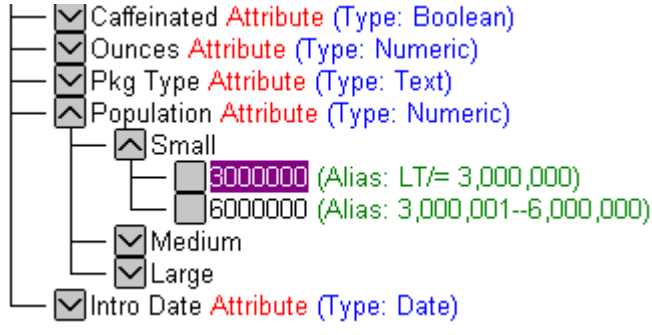
```
SEGNAME=BASIC, SERVER=edasol29, DBNAME=Basic, APPLNAME=Sample, $
 TIMEDIM=Year, $
 MEASURE=Measures, $
 MEMBER=COGS, AGGREGATE=NO, $
 MEMBER=TOTAL_EXPENSES, AGGREGATE=NO, $
 MEMBER=PROFIT_%, AGGREGATE=NO, $
 MEMBER=PROFIT_PER_OUNCE, AGGREGATE=NO, $
  SHARE=300-30, PARENT=Diet, DIM=Product, $
  SHARE=200-20, PARENT=Diet, DIM=Product, $
  SHARE=100-20, PARENT=Diet, DIM=Product, $
```

Example: Reporting Against Attribute Tagged Dimensions

This example uses data defined in the sample Master and Access Files in *Generating Attribute Dimensions* on page 631.

Note that using an Attribute Dimension as a BY field in a request is only supported with the MDX adapter (ENGINE ESSBASE SET MDX ON).

The following request references the Population attribute dimension in the ATTR segment of the Master File. The request also references a member of the Population attribute dimension, 3000000. Market is the base dimension for the Population attribute dimension.



The BY phrase in the request references STATE, which falls within the 3rd Generation (GEN=3) of the Market dimension. This reference is consistent with the following image in the Essbase outline.

The outline names the Base dimension (Market) and identifies the generation (GEN3) that you must reference within that Base dimension when you create an Essbase request using the specified member (3000000) of the Population attribute dimension.

```
TABLE FILE BASIC
PRINT DATA_VALUE
BY STATE
WHERE POPULATION EQ '3000000'
END
```

The output is:

```
STATE           DATA_VALUE
-------------   ----------
Iowa              9,061.00
Nevada            4,039.00
New Hampshire     1,125.00
New Mexico          330.00
Utah              3,155.00
```

The next request generates a message because the generation of the referenced field, REGION, falls within the 2nd generation of the base dimension, Market, as shown in the previous outline.

```
TABLE FILE BASIC
PRINT DATA_VALUE
BY REGION
WHERE POPULATION EQ '3000000'
END
```

As a result, the request displays the following message:

```
(FOC43271) Small_3000000 is not an associated attribute of any requested
column
```

Describing Measure Groups in the Master File

Measure groups in Essbase can be represented in two ways to the server: as normal dimensions or as pseudo accounts dimensions. If they are to be described as normal dimensions, the synonym creation process generates field descriptions for the dimension. If you wish to describe them as pseudo accounts in order to generate the measure group in place of the accounts dimension segment, use the SET SCENARIO command. You must issue the SET SCENARIO command before the synonym is created.

Syntax: How to Generate a Measure Group

If you wish to generate the Scenario dimension to be used in place of the accounts dimension segment, issue

```
ENGINE ESSBASE SET SCENARIO DIM dimension_name [member_name|ALL] FOR
synonym
```

where:

dimension_name

Is the measure group name.

When used in the SET SCENARIO command, *dimension_name* is case-sensitive and must match the case in the Essbase outline.

member_name

Specifies a member name in the measure group to be used to generate a field for every measure intersection. When used in the SET SCENARIO command, *member_name* is case-sensitive and must match the case in the Essbase outline.

ALL

Indicates that all members of the measure group are used to generate the Master File. ALL is the default value.

synonym

Is the Master File name for one *application.database* combination.

Note: You can issue the SET SCENARIO command multiple times to specify that a number of members from the measure group be used in the generation of the Master File. Once this command has been issued, you can create a synonym to generate the Master File name.

For each Measure group/Accounts member intersection, a field is generated in the Master File. If this multiplicity effect causes the Master File that is generated to be invalid (due to the total length of the fields), synonym creation fails and displays the following message:

```
Total actual or usage exceeds 32768 To cut down,try SET SCENARIO
```

If you receive this message, you will need to issue fewer SET SCENARIO commands to limit the scope of the Master File with regard to the number of measure groups used.

Describing the Measures Dimension in the Master File

The SET MEASURE command enables you to set or change the Accounts tagged dimension without having to change the outline in the Essbase Database Server. With this setting, the adapter produces an actual field name for every member of the named dimension in the Master File rather than producing the dimension in terms of generations. You must issue the SET MEASURE command before the synonym is created.

Syntax: How to Set or Change the Measures Dimension in the Master File

To set the Accounts tag for a dimension, issue

```
ENGINE ESSBASE SET MEASURE dimension_name FOR
synonym
```

where:

dimension_name

Is the name of the dimension to be interpreted as an Accounts tagged dimension when generating a synonym.

Generates a Master File in which the Accounts Tagged dimension is represented as generations.

synonym

Is the Master File name for one *application.database* combination.

Example: Using the SET MEASURE Command

The following command displays the actual member names of the measure group (for ACTUAL, BUDGET, or VARIANCE), rather than displaying fields like SCENARIO or GEN2_SCENARIO, in the Master File.

```
ENGINE ESSBASE SET MEASURE Scenario FOR SAMPLE
```

Reference: Limiting the Number of Fields in a Master File

SET MEASURE can be set to NONE. With this setting, all of the dimensions are interpreted as non-Accounts dimensions in the Master File and represented as generations.

In addition, a segment called DATA is added to the Master File. This segment contains a field called DATA_VALUE, which enables you to display the values of the Measures dimension although the actual member names are not present in the Master File.

Syntax: How to Limit the Number of Fields in a Master File

```
ENGINE ESSBASE SET MEASURE NONE FOR SAMPLE
```

where:

NONE

Generates a Master File in which the Accounts tagged dimension is represented as generations.

Example: Reporting Against a Master File With a Limited Number of Fields

If SET MEASURE has been set to NONE, you can display the values of the Measures dimension although the actual member names are not present in the Master File. The following is a sample DATA segment, which includes the field DATA_VALUE, against which you can report.

```
$  DIMENSION: DATA
  SEGMENT=DATA, SEGTYPE=U, PARENT=BASIC, $
    FIELDNAME=DATA_VALUE, ALIAS=DATA_VALUE, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, TITLE='DATA_VALUE', $
TABLE FILE BASIC
PRINT DATA_VALUE
BY PRODUCT
END
```

The output is:

```
PRODUCT             DATA_VALUE $
-------------       -------------------
Product             105,522.00
```

Changing the Default Usage Format of the Accounts Dimension

All dimension descriptions in the Master File, except for the accounts dimension, have the USAGE format of alphanumeric. The default format of the accounts dimension is D20.2. You can change this default using the SET CONVERSION command.

Syntax: How to Change the Default Usage Format of the Accounts Dimension

```
ESSBASE SET CONVERSION format PRECISION n m
```

where:

format

Possible values are:

FLOAT which indicates that the command applies only to double precision floating point columns.

DECIMAL which indicates that the command applies only to decimal columns.

n

Is the precision. It must be a valid number representing the maximum value for precision for the data type.

m

Is a valid number representing the maximum value for scale for the data type.

If you do not specify a value for scale, the current scale setting remains in effect. If the scale is not required, you must set *m* to 0 (zero).

Customizing the Essbase Environment

The Adapter for Essbase provides several parameters for customizing the environment and optimizing performance. The following topics provide an overview of customization options.

Using the MDX Adapter

Essbase Version 7 makes an interface available that communicates with the Essbase engine using MDX instead of the Essbase proprietary query language. If you issue the SET MDX ON command in any supported profile, the Adapter for Essbase will submit queries to Essbase using MDX.

The MDX version of the Adapter for Essbase provides the following capabilities:

- There is no default limit or maximum for the number of rows returned by a query. Therefore, the MAXROWS setting is not needed.
- You gain additional functionality for attribute dimensions, which are groups of properties that apply to a specific level-based field. An attribute dimension is stored as a flat hierarchy (all of the members are stored on a single level immediately below the dimension name in the synonym).

  Attribute dimensions can be used as filters in a query whether it is based on level hierarchies or parent/child hierarchies. In order to use the not-equal operator with an attribute dimension, you must use a WHERE TOTAL phrase rather than a WHERE phrase.

  With a level hierarchy, an attribute dimension can also be used as a BY field, as long as the field to which it is linked is also a BY field in the request.

Syntax: How to Enable the MDX Adapter for Essbase

```
ENGINE ESSBASE SET MDX {ON|OFF}
```

where:

`ON`

Enables the MDX adapter.

`OFF`

Uses the native adapter for Essbase. OFF is the default value.

Specifying ALIAS Names

In Essbase, you can assign more than one name to a member or a shared member. For example, in the *Sample.Basic* database, the PRODUCT dimension contains members that can be identified by both product codes, such as 200, or by more descriptive names, such as COLA. By default, the output values for members specified in a request are the member values. The server allows you to take advantage of the more descriptive member names in the Essbase outline when formulating a request.

Syntax: How to Specify ALIAS Names in a Request

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET USEALIASNAME {ON|OFF} [FOR synonym]
```

where:

`ON`

Allows the use of ALIAS names in a request.

`OFF`

Does not allow the use of ALIAS names in a request. OFF is the default value.

`synonym`

Is the Master File name for one *application.database* combination.

Note: If the SET USEALIASNAME command is used without FOR *synonym*, the setting applies to all Master Files. If the *synonym* option is used, the setting applies only to that Master File. You can issue multiple commands to control the use of the ALIAS name at the Master File level.

In an Essbase script, the reporting keyword OUTALTNAMES is equivalent to ALIAS.

Specifying ALIAS and Member Names in One Request

Procedure: How to Specify ALIAS and Member Names in a Request

When creating the synonym, check the *Make alias fields* check box to create a Master File that contains an alias field for each field in the generation view. For information on creating synonyms, see *Creating Synonyms* on page 620.

Example: **Generated Master File basic.mas**

```
FILENAME=SAMPLE, SUFFIX=ESSBASE, $
SEGMENT=BASIC, SEGTYPE=S0, $
.
.
.
 $  DIMENSION: Product
 DIMENSION=Product,CAPTION=Product, $
  HIERARCHY=Product,CAPTION=Product Levels,
      HRY_DIMENSION=Product,HRY_STRUCTURE=S, $
    FIELDNAME=PRODUCT, ALIAS='Lev2,Product', USAGE=A7, ACTUAL=A7,
      WITHIN='*Product', $
    FIELDNAME=FAMILY, ALIAS=Family, USAGE=A4, ACTUAL=A4,
      WITHIN=PRODUCT, $
    FIELDNAME=FAMILY_ALIAS, ALIAS=Family_ALIAS, USAGE=A11, ACTUAL=A11,
      REFERENCE=FAMILY, PROPERTY=CAPTION,  $
    FIELDNAME=SKU, ALIAS=SKU, USAGE=A6, ACTUAL=A6,
      WITHIN=FAMILY, $
    FIELDNAME=SKU_ALIAS, ALIAS=SKU_ALIAS, USAGE=A18, ACTUAL=A18,
      REFERENCE=SKU, PROPERTY=CAPTION,  $
```

Note: The SET USEALIASNAME ON/OFF command is ignored when the Master File is created using this option.

For information about the SET USEALIASNAME command, see *Specifying ALIAS Names* on page 639.

Using ALIAS and Member Names in an Essbase Request

```
TABLE FILE BASIC
PRINT SALES PROFIT_K
BY SKU
BY SKU_ALIAS
WHERE SKU_ALIAS EQ 'Cola'
END
```

The output is:

```
SKU      SKU_ALIAS   Sales      Profit
---      ---------   -------    --------
100-10   Cola        62,824.00  22,777.00
```

Specifying ALIAS Tables

In Essbase, aliases are generally stored in one or more tables as part of the database outline. Once the SET command is active, screening conditions must use the ALIAS value, not the member value.

If multiple ALIAS tables exist for an outline, you can use the SET ALIASTABLE command to specify which ALIAS table to use for a query.

Syntax: How to Specify an ALIAS Table

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET ALIASTABLE {aliastablename|RESET} FOR synonym
```

where:

aliastablename

Is the ALIAS table name to be used.

RESET

Sets the alias table back to the current default for the outline in Essbase.

synonym

Is the Master File name for one *application.database* combination.

Note: This SET ALIASTABLE command is used in conjunction with the SET USEALIASNAME command. If that command is not issued, the SET ALIASTABLE command is ignored. For more information, see *Specifying ALIAS Names* on page 639.

Setting the Maximum Number of Rows Returned

The default number of rows that can be returned from an Essbase cube using the server is 10,000. This can be controlled by the MAXROWS setting, which can be set in any supported server profile.

Note: The MAXROWS setting does not apply to the MDX Adapter, which has no default limit and no maximum number of rows returned.

Syntax: How to Set the Maximum Number of Rows Returned

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

To limit or extend the number of rows returned, use the syntax

```
ENGINE ESSBASE SET MAXROWS n [FOR synonym]
```

where:

n

Is a valid number that either limits or extends the number of rows to be returned. The default number of rows returned without this setting is 10,000. The maximum number for this setting is 999999999.

synonym

Is the Master File name for one *application.database* combination.

Time Series Reporting

By using Time Series and Accounts tags within your Essbase outline, you can tell Essbase how to calculate your accounts data. When you tag a dimension as Time, Essbase knows that this is the dimension on which to base the time periods for the Accounts tags.

The server supports all eight levels of period-to-date reporting within Essbase. See the *Hyperion Essbase Database Administrator's Guide* for further information.

In order to retrieve values from members in an Accounts dimension using time-series reporting, you must refer to at least one member of the Time dimension using the WHERE clause.

Example: Using Time Series Reporting

```
TABLE FILE BASIC
PRINT PRODUCT Q_T_D PROFIT_K AS 'Profit'
WHERE Q_T_D EQ 'Mar'
END
```

Where PROFIT_K is a member of the Accounts dimension, PRODUCT is a member of a non-Accounts dimension, and Q_T_D is a member of the Time dimension. The above request returns the following row:

```
Product Q-T-D(Mar)            24703.00
```

Note: All results reference the Dynamic Time series member with dashes between letters.

The equivalent request using SQL is:

```
SELECT Q_T_D, PRODUCT, PROFIT_K FROM BASIC WHERE Q_T_D = 'Mar'
```

Summing on Non-Aggregated Fields

The Essbase outline can contain two-pass calculated members, as well as members with different consolidation properties. Any two-pass calculated members that are found in the accounts (or Scenario) dimension members with the (-), (\), (*), and (%) consolidation properties in the Essbase outline, have the following attributes in the associated Access File:

```
MEMBER=membername, AGGREGATE=NO, $
```

By default (when Aggregation is ON), you cannot use the SUM command on two-pass calculated members or on members with the (-), (\),(*), and (%) consolidation properties. If a request contains a SUM action that uses one of these members, the following error message is displayed:

```
(FOC43241) Aggregation is requested for non aggregatable member(s)
```

However, when Aggregation is turned OFF, the server allows SUM to be used on two-pass calculated members and members with the (-), (\),(*), and (%) consolidation properties.

Syntax: How to Turn Aggregation ON/OFF For Non-Aggregated Fields

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET RESTRICTSUM [ON|OFF] FOR synonym
```

where:

`ON`

Does not allow the use of SUM on non-aggregated fields. ON is the default value.

`OFF`

Allows the use of SUM on non-aggregated fields.

`synonym`

Is the Master File name for one *application.database* combination.

Example: Using RESTRICTSUM on Non-Aggregated Fields

Using this Access File

```
SEGNAME=BASIC, SERVER=unxsol26, DBNAME=Basic, APPLNAME=Sample, $
TIMEDIM=Year, $
MEASURE=Measures, $
MEMBER=COGS, AGGREGATE=NO, $
MEMBER=TOTAL_EXPENSES, AGGREGATE=NO, $
MEMBER=PROFIT_%, AGGREGATE=NO, $
MEMBER=PROFIT_PER_OUNCE, AGGREGATE=NO, $
```

issue the following request:

```
TABLE FILE BASIC
SUM COGS
BY QUARTER
END
```

If RESTRICTSUM ON (default) is set in the profile, the following message is displayed and no output is generated:

```
(FOC43241) Aggregation is requested for non aggregatable member(s)
```

If you set RESTRICTSUM OFF in the profile, SUM is permitted on non-aggregated fields, producing this output:

```
QUARTER                    COGS
-------                 ---------
Qtr1                    42,877.00
Qtr2                    45,362.00
Qtr3                    47,343.00
Qtr4                    43,754.00
```

Preventing Aggregation of Non-Consolidating Members

Member consolidation properties determine how children roll up into their parents in the Essbase outline. The members with the (~) as a consolidation property in the Essbase outline, are *not* rolled up in the database. The SET AGGREGATE NOOP (No Operation) setting only affects those Essbase members in the Accounts dimension with the (~) consolidation property. If AGGREGATE NOOP is set to OFF, SUM is not permitted on Essbase members with the (~) consolidation property and those members appear in the Access File as:

```
MEMBER=ADDITIONS, AGGREGATE=NO, $
```

Syntax: How to Prevent Aggregation of Non-Consolidating Members

You must turn set AGGREGATE NOOP to OFF before you create the synonym.

```
ENGINE ESSBASE SET AGGREGATE NOOP {ON|OFF}
```

where:

ON

Allows the use of SUM on NOOP Essbase members. With this setting, Essbase members are not added to the Access File. ON is the default value.

OFF

Does not allow the use of SUM on NOOP Essbase members. Members with the (~) consolidation property are added to the Access File.

Note: Any member of the Accounts dimension in the Essbase outline tagged with (-), (/), (*), or (%) as a consolidation property automatically appears in the Access File with the AGGREGATE=NO setting.

Suppressing Shared Members

Shared Members in Essbase store pointers to data that is stored in the real member. Therefore, although the data is shared between the two members, it is only stored one time. You can exclude shared members from reports using the SUPSHARE setting.

Syntax: How to Suppress Shared Members in Essbase

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET SUPSHARE [ON|OFF]
```

where:

`ON`

Excludes shared members in a report.

`OFF`

Includes shared members in a report. OFF is the default value.

Example: Suppressing Shared Members

The following request and output illustrates the affect of turning SUPSHARE OFF and ON.

```
TABLE FILE SAMPLE
PRINT SALES BY FAMILY BY SKU
AND COLUMN-TOTAL
WHERE FAMILY EQ 'Diet' OR '200'
END
```

With SUPSHARE OFF, the output is:

```
FAMILY  SKU         SALES
------  ------  ---------
200     200-10  41,537.00
        200-20  38,240.00
        200-30  17,559.00
        200-40  11,750.00
Diet    100-20  30,469.00
        200-20  38,240.00
        300-30  36,969.00
=========================
TOTAL          214,764.00
```

With SUPSHARE ON, the output is:

```
FAMILY  SKU         SALES
------  ------  ---------
200     200-10  41,537.00
        200-20  38,240.00
        200-30  17,559.00
        200-40  11,750.00
Diet    100-20  30,469.00
        300-30  36,969.00
=========================
TOTAL          176,524.00
```

Notice that member 200-20 is displayed twice in the first report, but only once in the second report. Turning SUPSHARE ON prevents a shared member from being duplicated in a report request.

Suppressing Zero Values

Essbase stores zero values that you can exclude from reports using the SUPZEROS setting. This setting applies to an entire row. If one member in a row has a value of zero, and the other values in the same row have values other than zero, the zero value will not be suppressed.

Syntax: How to Suppress Rows With Zero Values

```
ENGINE ESSBASE SET SUPZEROS [ON|OFF]
```

where:

`ON`

Suppresses an entire row that contains 0 values.

`OFF`

Does not suppress zero values. OFF is the default value.

Example: Suppressing Zero Values

The following output illustrates the affect of turning SUPZEROS OFF and ON.

With SUPZEROS OFF, the output is:

```
STATE           SALES      PROFIT
---------    ---------   ---------
Colorado    38,240.00   42,877.00
Louisiana   0.00        0.00
Nevada      29,156.00   47,343.00
```

With SUPZEROS ON, the output is:

```
STATE           SALES     PROFIT
--------    ---------  ---------
Colorado    38,240.00  42,877.00
Nevada      29,156.00  47,343.00
```

Suppressing Missing Data

In Essbase, empty cells are known as missing or #MISSING data. You can suppress missing data during reporting using the SUPMISSING setting.

Syntax: How to Suppress Rows With Missing Data

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET SUPMISSING [ON|OFF]
```

where:

`ON`

Suppresses an entire row that contains #MISSING data.

`OFF`

Does not suppress rows with #MISSING data. OFF is the default value.

Example: Suppressing Missing Data

The following output illustrates the affect of turning SUPMISSING OFF and ON.

With SUPMISSING OFF, the output is:

```
STATE           SALES     PROFIT
---------   ---------  ---------
Colorado    38,240.00  42,877.00
Louisiana   #MISSING   #MISSING
Nevada      29,156.00  47,343.00
```

With SUPMISSING ON, the output is:

```
STATE           SALES     PROFIT
--------    ---------  ---------
Colorado    38,240.00  42,877.00
Nevada      29,156.00  47,343.00
```

Suppressing Zero Values and Missing Data

You can use the SUPEMPTY setting to suppress an entire row that contains either zero values, missing (#MISSING) data, or a combination of the two.

Syntax: How to Suppress Rows With Zero Values or Missing Data

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET SUPEMPTY [ON|OFF]
```

where:

`ON`

Suppresses an entire row that contains either zero values, #MISSING data, or a combination of the two.

`OFF`

Does not suppress rows that contain either zero values or #MISSING data. OFF is the default value.

Example: Suppressing Rows With Zero Values or Missing Data

The following output illustrates the affect of turning SUPEMPTY OFF and ON.

With SUPEMPTY OFF, the output is:

```
STATE           SALES     PROFIT
---------    ---------  ---------
Colorado     38,240.00  42,877.00
Louisiana    #MISSING   #MISSING
Nevada       0.00       0.00
```

With SUPEMPTY ON, the output is:

```
STATE           SALES     PROFIT
--------     ---------  ---------
Colorado     38,240.00  42,877.00
```

Substitution Variables

Substitution variables act as global placeholders for information that changes regularly. Each substitution variable configured in Essbase is assigned a value. You can change the value at any time, thus limiting the number of manual changes required in your scripts when you are running reports.

When you reference an Essbase substitution variable in a TABLE command, the substitution variable must be preceded with an up arrow (^) and enclosed in single quotation marks.

Example: Using a Substitution Variable in a Request

In this example, CurrMonth is the name of the substitution variable that was set in Essbase. The ^ and quotation marks are required.

```
TABLE FILE BASIC
PRINT Product
BY
PRODUCT
WHERE
MONTH EQ '^CurrMonth'
END
```

Using the SPARSE Data Extraction Method

The SET SPARSE ON command tells Hyperion Essbase to use the sparse data extraction method, which optimizes performance when a high proportion of the reported data rows are #MISSING. This data extraction method is different from the regular method. Hyperion Essbase cannot use the sparse data retrieval optimization method on Dynamic Calc or Dynamic Calc And Store members.

Syntax: How to Use the SPARSE Data Extraction Method

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE ESSBASE SET SPARSE {ON|OFF}
```

where:

`ON`

Uses the sparse data extraction method when a high proportion of the reported data rows are #MISSING.

`OFF`

Does not use the sparse data extraction method. OFF is the default value.

Setting a Default Connection

Once connections have been defined in the edasprof.prf server global profile, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection, which is used if the Access File does not specify connection name or server. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE ESSBASE SET DEFAULT_CONNECTION connection
```

where:

`ESSBASE`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE ESSBASE SET DEFAULT_CONNECTION SAMPLE
```

Supporting Unicode Mode Applications

An Essbase application can be set to work in Unicode or non-Unicode mode. If the Reporting Server is configured for Unicode, no setting is required to use a cube set for Unicode mode in a Unicode mode application. However, if the server is not configured for Unicode but your Essbase cube is configured for Unicode, you must issue the ENGINE ESSBASE SET UNICODE ON command.

Syntax: How to Configure the Adapter for Unicode Mode Applications

```
ENGINE ESSBASE SET UNICODE {OFF|ON}
```

where:

OFF

Does not support Unicode mode applications when the server is not configured for Unicode. OFF is the default value.

ON

Supports Unicode mode applications when the server is not configured for Unicode.

Syntax: How to Determine Whether Unicode Mode Applications Are Supported

```
ENGINE ESSBASE SET ?
```

When you issue this command, the following message indicates whether Unicode mode is on for your Essbase cube:

```
(FOC43249) Essbase Unicode Option : on
```

Essbase Reporting With WebFOCUS

Overview of Essbase Reporting Concepts

In a multi-dimensional data source (cube), dimensions are categories of data, such as Region or Time, that you use to analyze and compare business performance. Dimensions consist of data elements that are called *members*. For example, a Region dimension could have members *England* and *France*.

Dimension members are usually organized into hierarchies. Hierarchies can be viewed as tree-like graphs where members are the nodes.

For example, the Region dimension may have the element *World* at its top level (the root node). The World element may have children nodes (members) representing continents. Continents, in turn, can have children nodes that represent countries, and countries can have children nodes representing states or cities. Nodes with no children are called *leaf nodes*.

Measures are numeric values, such as Sales Volume or Net Income, that are used to quantify how your business is performing.

A multi-dimensional cube consists of data derived from facts, which are records about individual business transactions. For example, an individual fact record reflects a sales transaction of a certain number of items of a certain product at a certain price, which occurred in a certain store at a certain moment of time. The cube contains summarized fact values for all combinations of measures and members of different dimensions.

For example, the following combination (*tuple*) contains the total volume of sales of pumps in all stores in England in 2005:

```
{Sales Volume, Pumps, England, 2005}
```

The point in the cube that contains this summarized value is called a cell. A cell is addressed by a combination of members of different dimensions and a measure. In this example *Sales Volume* is a measure and *Pumps*, *England*, and *2005* are members of the Product, Region, and Time dimensions respectively.

Individual fact records are usually tied to the leaf nodes of each hierarchy in the cube. The fact values get included in cells addressed by these leaf nodes and added to all cells addressed by all combinations of ascendants of these leaf nodes along each hierarchy of the cube.

The operation used to summarize facts for some measures can be a simple sum or a more complex aggregation function such as an average.

In Essbase, it is assumed that a value exists for each combination of hierarchy values contained in the cube. Measures are not represented separately until the metadata is created. At that time you have the choice of simply creating a "cell" value tied to the existing combinations or exposing one or two of the dimensions as the measures.

Some combinations of hierarchy nodes may not have any fact records tied to them. The cells addressed by these combinations are empty cells.

As illustrated in the previous example, a tuple is a combination of members from different dimensions of a cube. The previous tuple contains members from all dimensions and, therefore, addresses a single cell. If a tuple contains only members from some dimensions, it addresses not just one cell but a whole slice of cells in the cube. For example, the following tuple does not include either Region or Time dimension members:

```
{Sales Volume, Pumps}
```

It addresses as many cells in the cube as there are members of the Region dimension times the number of members of the Time dimension.

The number of cells in the cube addressed by a single dimension member is a product of cardinalities of all other dimensions.

When all cells addressed by a member are empty, the member is called an empty member. When all cells addressed by a tuple are empty, the tuple is called an empty tuple.

By default, empty cells appear on report output. You can issue the following command if you want to eliminate them on the report output:

```
ENGINE ESSBASE SET SUPMISSING ON
```

Cubes can also include two types of attributes, which are parameters used in data selection. Attributes cannot be displayed or used for sorting; they can only be used in selection criteria.

When the adapter accesses a multi-dimensional cube, it uses two types of metadata elements about dimensions:

- **Hierarchy fields.** These fields contain data that apply to a specific hierarchy and describe the position of each member within that hierarchy. For example, these fields identify the parent of the member, as well as its own name, caption, and unique ID.
- **Dimension properties.** These fields contain data that potentially apply to all members of the dimension. For example, a Region dimension may have a property called GEOGRAPHICAL_HEIGHT that specifies the altitude for each member of the Region dimension.

Using the hierarchy fields, the adapter can recreate the hierarchy and locate portions of the hierarchy needed to satisfy a request.

Understanding Columnar and Hierarchical Reporting

Two types of hierarchy are represented in a synonym: level and parent/child.

A synonym describes a level hierarchy by using a separate field for each level. To report on a level hierarchy, you use columnar reporting in which you specify the field name for each level you want to display.

A synonym describes a parent/child hierarchy using a set of fields that define the hierarchical structure and the relationships between the hierarchy members. The adapter has special hierarchical reporting syntax for reporting on parent/child hierarchies.

Hierarchical reporting enables you to sort and select members of parent/child hierarchies without knowing specific generation numbers. A hierarchical reporting request goes through several phases before output is displayed.

Hierarchical Sorting and Member Selection

The first phase selects hierarchy members to display. The hierarchical reporting phrase BY or ON HIERARCHY automatically sorts and formats a hierarchy with appropriate indentations that show the parent/child relationships. If you do not want to see the entire hierarchy, you can use the WHEN phrase to select hierarchy members for display. The expression in this WHEN phrase must reference only hierarchy fields, not dimension properties or measures.

Measures and dimension properties are linked to the leaf nodes of the dimension and, therefore, cannot be used in selecting hierarchy levels for display.

Screening Dimension Data

Once hierarchy members are selected, you can screen the retrieved dimension data by applying WHERE tests to the selected members.

WHERE criteria are applied to the leaf nodes and are processed after the phase of the request that selects hierarchy members. Therefore, dimension properties can be used in WHERE tests.

These tests can also reference hierarchy fields. However, since the selection criteria are always applied to the values at the leaf nodes, they cannot select data based on values that occur at higher levels. For example, in a dimension with Continents, Countries, and Cities, your request will not display any rows if you use WHERE to select at the Country level, but it may if you use it to select at the City level.

WHERE tests can also reference measures if the ENGINE ESSBASE RESTRICTSUM OFF command is in effect.

Screening Based on Aggregated Values

Measures, being summarized values, can be referenced in WHERE TOTAL tests and COMPUTE commands because those commands are processed after the hierarchy selection and aggregation phases of the request. When screening with WHERE TOTAL, the aggregation phase of the report processing is over, so totals on the report are not recalculated to account for the data that is screened out, the rows are just removed. In addition, if you use the RESTRICTSUM OFF setting, you can apply DEFINEs and WHERE tests to measures.

Reference: Prerequisites for Hierarchical Reporting

Hierarchical reporting uses special metadata attributes and reporting syntax. You must:

- ❑ Create a synonym that describes hierarchies as parent/child relationships.

- Use hierarchical reporting syntax in your request to automatically build and format hierarchies in the report output.

For more information on creating synonyms, see *Creating Synonyms* on page 620. For more information on hierarchical reporting, see *Hierarchical Reporting* on page 659.

Representing Hierarchies in a Synonym

Dimensions are organized into sets of hierarchies. For example, in a Time dimension, years, quarters, and months can form a hierarchical or parent/child relationship. This means that the measures for each month are aggregated into values for quarters, and the quarters are aggregated into values for years. Each point in the hierarchy is called a *node*. Nodes at the bottom of the hierarchy (with no children) are called *leaf nodes*.

In a synonym, hierarchies can be described in one of two ways:

- **Level hierarchy.** Each hierarchy level is described using a separate field name. To issue a report request, you must reference the field name for those levels you want to display on the report output.

  When you create a synonym, the synonym contains one field declaration for each level. This declaration also specifies which field is its parent. If the data changes to have an additional level, you must recreate the synonym in order to account for this additional field and parent reference.

- **Parent/child hierarchy.** The hierarchy is described with a set of fields that contain values for properties that describe the position of each member in the hierarchy. For example, there are fields to contain the unique ID, parent, and caption of a member. To issue a report request, you only need to specify the field name of one of the hierarchy fields.

  With a parent/child hierarchy, one set of field names in the synonym describes the hierarchy. A change in the number of levels does not require a change to the parent/child representation of the hierarchy in the synonym.

  Another advantage of parent/child hierarchies is that the adapter can recreate and format any portion of the hierarchy. The request does not have to specify generation numbers.

  Both level and parent/child hierarchies are created for each dimension in the synonym.

Dimension properties apply to all hierarchies in the dimension and are listed in the synonym following all of the hierarchies.

Example: Dimension Declaration

Each dimension begins with a dimension record that defines the dimension and its hierarchies. The dimension itself is generation zero.

Dimension:

```
DIMENSION=Products, CAPTION='Products', $
```

Example: Describing a Level Hierarchy

Each generation of the hierarchy is assigned a field name consisting of the hierarchy name (for example, Manufacturing) with the generation number appended (unless you changed the generation names in the Essbase Client application). Each field declaration also specifies the field name of its parent with the WITHIN attribute. The value stored in this field is the member caption (title).

If a new generation appears in the data, the synonym must be recreated to define this new generation.

The following field describes generation three. Its parent is generation 2:

```
FIELDNAME=WAREHOUSE, ALIAS=Warehouse, USAGE=A11, ACTUAL=A11,
     WITHIN=PLANT,
     PROPERTY=UID,  $
```

Example: Describing a Parent/Child Hierarchy

Several fields are used to define a parent/child hierarchy. Each has a PROPERTY attribute that describes which hierarchy property it represents. The hierarchy field names are formed by appending a suffix to the hierarchy name.

For example, the caption of a hierarchy named Manufacturing2 is stored in a field whose name is MANUFACTURING_CAPTION and whose property attribute is PROPERTY=CAPTION.

The following table describes the hierarchy fields:

Description of Data	PROPERTY=	Field Suffix
Member's Unique ID (unique within the cube)	UID	_MEMBER
Member's Level Number	LEVEL_NUMBER	(Currently not used)
Member's Parent	PARENT_OF	_PARENT
Member's Caption (title on reports)	CAPTION	_CAPTION
Parent's Caption	CAP_PARENT	_PARENTCAP

The following declaration for the Manufacturing2 hierarchy describes the field that contains a member's unique ID (PROPERTY=UID):

```
FIELDNAME=MANUFACTURING_MEMBER, ALIAS=Manufacturing, USAGE=A13,
     ACTUAL=A13,
     WITHIN='*Manufacturing2',
     PROPERTY=UID,  $
```

The following declaration for the Products2 hierarchy defines the field that contains a member's title (PROPERTY=CAPTION):

```
FIELDNAME=MANUFACTURING_CAPTION, USAGE=A17, ACTUAL=A17,
     REFERENCE=MANUFACTURING_MEMBER, PROPERTY=CAPTION,  $
```

Example: Dimension Properties

Following all of a dimension's hierarchies, the dimension properties (called *attributes*) are described. Each of these has PROPERTY=UDA (User Defined Attribute) in the synonym.

For example the following field represents a member's warehouse management property:

```
FIELDNAME=WHSE_MANAGEMENT, ALIAS='Whse Management', USAGE=A15,
   ACTUAL=A15,
   REFERENCE=WAREHOUSE, PROPERTY=UDA,  $
```

Example: Sample Request Using a Level Hierarchy

A report request against a level hierarchy must specify the field name for each generation of the hierarchy required in the report.

For example, the following request displays the sales dollars measure for generations one through three of the Manufacturing hierarchy:

```
TABLE FILE CENTSALE
SUM SALES_DOLLARS
BY MANUFACTURING AS 'LEVEL1'
BY PLANT AS 'LEVEL2'
BY WAREHOUSE AS 'LEVEL3'
ON TABLE COLUMN-TOTAL
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

LEVEL1	LEVEL2	LEVEL3	Sales Dollars
Manufacturing	PLANT 01	Whse 2	129,327,743.98
		Whse 8	130,976,032.31
		Whse 9	132,413,208.98
	Plant 02	Whse 1	125,547,866.13
		Whse 3	139,263,132.99
		Whse 4	129,286,784.49
		Whse 6	210,270,945.82
	Plant 03	Whse 5	131,243,866.46
		Whse 7	122,413,709.11
TOTAL			1,250,743,290.27

In order to get a total for the entire hierarchy, you have to use an ON TABLE COLUMN-TOTAL command in the request or add another SUM command without a BY phrase (and this would add another column to the report output).

Example: Sample Request Using a Parent/Child Hierarchy

A report request against a parent/child hierarchy can use the BY HIERARCHY phrase to report against the entire hierarchy. The output is automatically formatted with appropriate indentations to show the hierarchy generations and relationships.

For example, the following request shows the sales volume measure for three generations of the Manufacturing2 hierarchy:

```
TABLE FILE CENTSALE
SUM SALES_DOLLARS
BY MANUFACTURING_CAPTION HIERARCHY
SHOW TO DOWN 3
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

MANUFACTURING_CAPTION	Sales Dollars
Manufacturing	1,250,743,290.27
Denver Plant	392,716,985.27
Denver	129,327,743.98
Salem	130,976,032.31
San Diego	132,413,208.98
Kansas City Plant	604,368,729.43
St Louis	125,547,866.13
Dallas	139,263,132.99
Kansas City	210,270,945.82
Seattle	129,286,784.49
Columbia Plant	253,657,575.57
Bangor	131,243,866.46
Columbia	122,413,709.11

The report request does not have to reference specific hierarchy generations. The BY HIERARCHY phrase recreates and formats the hierarchy for display. You can also use a WHEN phrase to select a portion of the hierarchy and a SHOW phrase to specify how many levels above and below the selected portion of the hierarchy you want to display. This display format clearly shows the parent/child relationships between the hierarchy members.

In addition, you can specify whether you want the measure values for each parent to represent aggregates for all of its children (full total) or only those selected for display (visual total).

For more information, see *Hierarchical Reporting* on page 659.

Hierarchical Reporting

When you issue a request against a cube, some of the requirements and features available depend on the type of hierarchy you are reporting against.

With a parent/child hierarchy, you can specify whether the measure values displayed for each parent should show the sum of all of its descendants (full total) or the sum of its displayed descendants (visual total). For level hierarchies, the report always displays full totals, which are the values actually stored in the cube.

For parent/child hierarchies, you can use the BY HIERARCHY phrase to sort and format the hierarchy. You can also limit the portion of the hierarchy selected for display using the WHEN phrase.

When a hierarchical request is processed, the first step is to build the hierarchy and mark which nodes should be included, which should be excluded, and which are needed for context.

The next stage fills the hierarchy with measure values. This stage applies WHERE criteria at the leaf nodes to further qualify the members selected for the report. Dimension properties cannot be used in the initial selection phase of the request, but can be used to screen the selected rows based on dimension data.

Measure values cannot be used to select hierarchy levels for reporting unless the ENGINE ESSBASE RESTRICTSUM OFF setting is in effect. After the hierarchy rows have been selected, screened, and aggregated, WHERE TOTAL tests can limit the rows displayed based on measure values.

It is not currently recommended to use hierarchical reporting with shared members. However, if the ENGINE ESSBASE SUPSHARE ON setting is in effect, a hierarchical report against a hierarchy with shared measures will include any specified shared members, but not members below those shared members.

Syntax: How to Display Parent/Child Hierarchies

The following syntax can be used to generate hierarchical reports when the synonym defines parent/child hierarchies:

```
SUM [FROLL.]measure_field ...
BY hierarchy_field [HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO {BOTTOM|DOWN m}] [byoption [WHEN condition] ...] ]
.
.
.
[WHERE expression_using_dimension_data]
.
.
.
[ON hierarchy_field HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO BOTTOM|DOWN m] [byoption [WHEN condition] ...]]
```

where:

FROLL

Specifies a full roll-up of the measure. With a full roll-up, the value displayed is the value found in the cube. This value may not reflect the sum of its displayed descendants if some descendants are eliminated from the output based on the WHEN and SHOW options. When FROLL is not specified, the value displayed is a visual total, which means it is the total of the values for its displayed descendants.

Note: If a request uses WHERE criteria to screen out some data, FROLL will not display the value found in the cube. It will display the roll-up of the selected data.

measure_field

Is the field name of a measure.

BY *hierarchy_field* HIERARCHY

Identifies the hierarchy used for sorting. The field must be a hierarchy field.

ON *hierarchy_field* HIERARCHY

Identifies the hierarchy used for sorting. The field must be a hierarchy field. The request must include either a BY phrase or a BY HIERARCHY phrase for this field name.

WHEN *expression_using_hierarchy_fields*;

Selects hierarchy members. The WHEN phrase must immediately follow the word HIERARCHY to distinguish it from a WHEN phrase associated with a BY option (such as SUBFOOT). Any expression using only hierarchy fields is supported. The WHEN phrase can be on the BY HIERARCHY command or the ON HIERARCHY command, but not both.

SHOW

Specifies which levels to show on the report output relative to the levels selected by the WHEN phrase. If there is no WHEN phrase, the SHOW option is applied to the root node of the hierarchy. The SHOW option can be specified on the BY HIERARCHY phrase or the ON HIERARCHY phrase, but not both.

n

Is the number of ascendants above the set of selected members that will have measure values. All ascendants appear on the report to show the hierarchical context of the selected members. However, ascendants that are not included in the SHOW phrase appear on the report with missing data symbols in the report columns that display measures. The default for *n* is 0.

TOP

Specifies that ascendant levels to the root node of the hierarchy will be populated with measure values.

TO

Is required when specifying a SHOW option for descendant levels.

BOTTOM

Specifies all descendants to the leaf nodes of the hierarchy will be populated with measure values. This is the default value.

m

Is the number of descendants of each selected level that will display. The default for *m* is BOTTOM, which displays all descendants.

byoption

Is one of the following sort-based options: PAGE-BREAK, REPAGE, RECAP, RECOMPUTE, SKIP-LINE, SUBFOOT, SUBHEAD, SUBTOTAL, SUB-TOTAL, SUMMARIZE, UNDER-LINE. If you specify SUBHEAD or SUBFOOT, you must place the WHEN phrase on the line following the heading or footing text.

condition

Is a logical expression.

expression_using_dimension_data

Screens the rows selected in the BY/ON HIERARCHY and WHEN phrases based on dimension data. The expression can use dimension properties, measures (with RETRICTSUM OFF), and hierarchy fields; however, the selection criteria are always applied to the values at the leaf nodes. Therefore, you cannot use WHERE to select rows based on hierarchy field values that occur at higher levels. For example, in a dimension with Continents, Countries, and Cities, your request will not display any rows if you use WHERE to select a Country name, but it may if you use it to select a City name.

The following examples illustrate hierarchical reporting using the BY HIERARCHY phrase.

Example: Reporting on a Whole Hierarchy

The following request produces visual totals (SALES_DOLLARS) and full totals (FROLL.SALES_DOLLARS). The BY HIERARCHY phrase specifies hierarchical reporting. There is no WHEN phrase to limit the portion of the hierarchy displayed. Also note that the full roll-up is displayed with the prefix FSUM added to the field name. The full total is the same as the visual total for a report on the entire hierarchy:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Reporting on a Whole Hierarchy"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The top portion of the hierarchy is:

Reporting on a Whole Hierarchy

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	1,594,324,089.21	1,594,324,089.21
STEREO	275,420,672.99	275,420,672.99
STEREO EQUIPMENT	275,420,672.99	275,420,672.99
MISC STEREO	197,208,300.45	197,208,300.45
PORTABLES	56,659,661.82	56,659,661.82
PERS. STER C/R- CASS	677,052.54	677,052.54
PERS. STER AM/FM CASSETTE PLAYER (Sony)	1,908,651.88	1,908,651.88
PERS. STER (Sony)	495,415.80	495,415.80
AM STEREO/FM STEREO PERS. STER	1,597,122.90	1,597,122.90
PERS. STER WITH MEGA BASS (Sony)	829,756.44	829,756.44
FM/AM PERS. STER	1,628,774.85	1,628,774.85
FM PERS. STER	2,071,231.36	2,071,231.36
PERS. STER AM/FM CASSETTE PLAYER (B-5)	249,689.24	249,689.24
CD PERS. STER (Sony)	1,089,902.07	1,089,902.07
SPORTS CD PERS. STER (Sony)	790,566.60	790,566.60
CD PERS. STER (Coby)	1,554,616.05	1,554,616.05
CD PERS. STER D-EJ360	2,200,685.70	2,200,685.70
CD PERS. STER (Magnavox)	1,845,510.69	1,845,510.69
PERS. STER (Zenith)	449,688.98	449,688.98
PERS. STER WITH MEGA BASS (Panasonic)	1,706,855.95	1,706,855.95
RECORDING AM/FM PERS. STER	586,146.00	586,146.00
PERS. STER TAPE PLAYER	447,224.05	447,224.05
PERS. STER AM/FM-	818,811.84	818,811.84
AM/FM CASSETTE PERS. STER W/AUTO SHUTOFF M	1,176,089.18	1,176,089.18

The bottom portion of the hierarchy is:

COMPUTER	171,594,840.43	171,594,840.43
COMPUTERS	171,594,840.43	171,594,840.43
LAP-DESKTOP	161,978,073.61	161,978,073.61
DESKTOP	50,508,621.77	50,508,621.77
ZEON DESKTOP	20,356,457.76	20,356,457.76
P4 DESKTOP COMPUTER	17,880,068.94	17,880,068.94
P4 DESKTOP	12,272,095.07	12,272,095.07
LAPTOP	111,469,451.84	111,469,451.84
NOTEBOOK COMPUTER 1.8	31,101,572.40	31,101,572.40
P4 LAPTOP	21,445,792.29	21,445,792.29
NOTEBOOK P-4=2.4/30GB/	24,296,762.60	24,296,762.60
LAPTOP COMPUTER 1.2	20,413,907.75	20,413,907.75
P4 NOTEBOOK COMPUTER 8KG	14,211,416.80	14,211,416.80

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
PDA	9,616,766.82	9,616,766.82
AMD MICRO	9,616,766.82	9,616,766.82
AMD DESKTOP	9,616,766.82	9,616,766.82

Example: Selecting a Hierarchy Member

The following request produces a visual total and full total for the Products2 hierarchy of the Products dimension, but limits the members selected for display with the WHEN phrase. Note that the hierarchy member selected is displayed in its context (all ancestors to the root of the hierarchy). However, the ancestors are not requested in the report and are, therefore, displayed with missing data symbols. All descendants of the selected members appear in the report output because the default SHOW option for descendants is BOTTOM:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'LAPTOP';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Hierarchy Member"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Hierarchy Member

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	.	.
COMPUTER	.	.
COMPUTERS	.	.
LAP-DESKTOP	.	.
LAPTOP	111,469,451.84	111,469,451.84
NOTEBOOK COMPUTER 1.8	31,101,572.40	31,101,572.40
P4 LAPTOP	21,445,792.29	21,445,792.29
NOTEBOOK P-4=2.4/30GB/	24,296,762.60	24,296,762.60
LAPTOP COMPUTER 1.2	20,413,907.75	20,413,907.75
P4 NOTEBOOK COMPUTER 8KG	14,211,416.80	14,211,416.80

Example: Selecting a Member and Adding a Parent

In the following request, the SHOW option UP 1 TO DOWN 0 added to the WHEN phrase adds the parent (LAP-DESKTOP) of the selected member (LAPTOP). This parent now contains values for the measures rather than missing data symbols. However, the full total column for the parent contains the sum of all of its descendants, not just the selected LAPTOP member, while the visual total shows the total only for the LAPTOP member:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'LAPTOP';
SHOW UP 1 TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding a Parent"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding a Parent

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	.	.
COMPUTER	.	.
COMPUTERS	.	.
LAP-DESKTOP	111,469,451.84	161,978,073.61
LAPTOP	111,469,451.84	111,469,451.84

Example: **Selecting a Member and Adding Children**

In the following request, the SHOW option UP 0 TO DOWN 1 added to the WHEN phrase adds the children of the selected member (LAPTOP). Because no children of the selected member are excluded, and no higher level members are in the SHOW set, the full and visual totals are the same:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'LAPTOP';
SHOW UP 0 TO DOWN 1
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding Children"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding Children

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	.	.
COMPUTER	.	.
COMPUTERS	.	.
LAP-DESKTOP	.	.
LAPTOP	111,469,451.84	111,469,451.84
NOTEBOOK COMPUTER 1.8	31,101,572.40	31,101,572.40
P4 LAPTOP	21,445,792.29	21,445,792.29
NOTEBOOK P-4=2.4/30GB/	24,296,762.60	24,296,762.60
LAPTOP COMPUTER 1.2	20,413,907.75	20,413,907.75
P4 NOTEBOOK COMPUTER 8KG	14,211,416.80	14,211,416.80

Example: Selecting a Member and Showing All Ascendants

In the following request, the SHOW option TOP TO DOWN 0 added to the WHEN phrase adds all ascendants but no descendants of the selected member (LAPTOP). These parents now contain values for the measures rather than missing data symbols. However, the full total column for the ascendants contains the sum of all of their descendants, not just the specified LAPTOP member, while the visual total shows the total for only the LAPTOP member:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'LAPTOP';
SHOW TOP TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding All Ascendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding All Ascendants

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	111,469,451.84	1,250,743,290.27
COMPUTER	111,469,451.84	65,905,435.61
COMPUTERS	111,469,451.84	65,905,435.61
LAP-DESKTOP	111,469,451.84	161,978,073.61
LAPTOP	111,469,451.84	111,469,451.84

Example: Selecting a Member and Showing All Ascendants and Descendants

In the following request, the SHOW option TOP added to the WHEN phrase adds all ascendants and descendants (since TO BOTTOM is the default) of the selected member (LAPTOP). These parents are now in the SHOW set and contain values for the measures rather than missing data symbols. However, the full total column for the ascendants contains the sums of all of their descendants, not just the specified LAPTOP member, while the visual total shows the total for only the LAPTOP member:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'LAPTOP';
SHOW TOP
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding All Ascendants and Descendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding All Ascendants and Descendants

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	111,469,451.84	1,250,743,290.27
COMPUTER	111,469,451.84	65,905,435.61
COMPUTERS	111,469,451.84	65,905,435.61
LAP-DESKTOP	111,469,451.84	161,978,073.61
LAPTOP	111,469,451.84	111,469,451.84
NOTEBOOK COMPUTER 1.8	31,101,572.40	31,101,572.40
P4 LAPTOP	21,445,792.29	21,445,792.29
NOTEBOOK P-4=2.4/30GB/	24,296,762.60	24,296,762.60
LAPTOP COMPUTER 1.2	20,413,907.75	20,413,907.75
P4 NOTEBOOK COMPUTER 8KG	14,211,416.80	14,211,416.80

Example: Selecting Members for Display and Screening on Member Values

You use the WHEN phrase to select hierarchy members to display on the report. Each of the selected members is displayed in its context. The ascendants of selected members display missing data symbols. When a request is processed, the first step is to build the hierarchy and mark which nodes should be included, which should be excluded, and which are needed for context but should display missing data symbols according to the WHEN phrase.

The next stage fills the hierarchy with measure values. This stage applies WHERE criteria (which must select on the lowest level of the hierarchy or the report will be empty).

If a WHERE is specified without a WHEN, no nodes are marked as excluded, so no nodes display missing values.

The following request shows the Manufacturing2 hierarchy:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_CAPTION HIERARCHY
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting Hierarchy Members"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting Hierarchy Members

MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
Manufacturing	1,250,743,290.27	1,250,743,290.27
Denver Plant	392,716,985.27	392,716,985.27
Denver	129,327,743.98	129,327,743.98
Salem	130,976,032.31	130,976,032.31
San Diego	132,413,208.98	132,413,208.98
Kansas City Plant	604,368,729.43	604,368,729.43
St Louis	125,547,866.13	125,547,866.13
Dallas	139,263,132.99	139,263,132.99
Kansas City	210,270,945.82	210,270,945.82
Seattle	129,286,784.49	129,286,784.49
Columbia Plant	253,657,575.57	253,657,575.57
Bangor	131,243,866.46	131,243,866.46
Columbia	122,413,709.11	122,413,709.11

The following request selects the portion of the hierarchy whose members contain the value *Plant* or *City*. It then screens on the dimension attribute WHSE_MANAGEMENT equal to *Leased* or *Owned*. This WHERE screening condition does not affect the portion of the hierarchy that was selected by WHEN, but it does not display or include in the totals those rows that had no data meeting the WHERE condition:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'City' OR 'Plant';
WHERE WHSE_MANAGEMENT EQ 'Leased' OR 'Owned'
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting Members and Screening Data Values"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting Members and Screening Data Values

MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
Manufacturing	.	.
Denver Plant	129,327,743.98	.
Denver	129,327,743.98	129,327,743.98
Kansas City Plant	394,097,783.61	.
St Louis	125,547,866.13	125,547,866.13
Dallas	139,263,132.99	139,263,132.99
Seattle	129,286,784.49	129,286,784.49
Columbia Plant	131,243,866.46	.
Bangor	131,243,866.46	131,243,866.46

Note that the full total is a value taken directly from the data source. It must pass both the WHEN and WHERE conditions in order to be retrieved and, therefore, is represented by a missing value in some rows. The visual total is calculated starting from the leaf nodes, so it is present for higher levels even if a higher level record does not pass the WHERE condition.

Example: Comparing Member Selection With Data Screening

The following request selects hierarchy members whose captions contains the value *Plant* and, from those selected members, screens values at the lowest level of the hierarchy for captions containing *City*:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'Plant';
WHERE MANUFACTURING_CAPTION CONTAINS 'City'
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Comparing Member Selection With Data Screening"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Comparing Member Selection With Data Screening

MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
Manufacturing	.	.
Kansas City Plant	210,270,945.82	210,270,945.82
Kansas City	210,270,945.82	210,270,945.82

Note that the WHERE condition removes all leaf nodes except *Kansas City* from the hierarchy display and from the totals.

Example: Reporting on Shared Members

The following request sets SUPSHARE ON and reports on the Customers2 hierarchy, which has shared members. The WHERE TOTAL test screens on the measure SALES_DOLLARS to remove rows with zero sales dollars. No members below the shared members display on the report output:

```
ENGINE ESSBASE SET SUPSHARE ON
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY CUSTOMERS_CAPTION HIERARCHY
WHEN CUSTOMERS_CAPTION CONTAINS 'Rep';
SHOW TO BOTTOM
WHERE TOTAL SALES_DOLLARS NE 0;
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Reporting on Shared Members"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Reporting on Shared Members

CUSTOMERS_CAPTION	Sales Dollars	FSUM Sales Dollars
Sales Representatives	707,840,214.13	1,250,743,290.27
Adam Scott	65,632,990.59	65,632,990.59
Andrew Lowell	.	148,809,230.28
Barbara Shas	.	244,107,115.97
Dan Joshes	958,086.61	958,086.61
Eric Craig	206,543,715.76	206,543,715.76
Justin Miller	.	63,672,254.93
Rich Green	1,966,189.11	1,966,189.11
Robyn Gold	421,507,869.83	421,507,869.83
Steven Davis	.	86,314,474.96
Tim Karis	11,231,362.23	11,231,362.23

Example: Using SHOW Without WHEN

The following request does not contain a WHEN phrase, so the SHOW option DOWN TO 0 applies to the root node (as if there had been a WHEN phrase that selected only the root node). Therefore, the root level is the only one shown on the report output:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
SHOW TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"SHOW Without WHEN"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

SHOW Without WHEN

PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Products	1,594,324,089.21	1,594,324,089.21

Example: Using SKIP-LINE

The following request adds the BY option SKIP-LINE to the BY HIERARCHY phrase:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'City';
SHOW UP 1 TO DOWN 1 SKIP-LINE
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using SKIP-LINE"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using SKIP-LINE

MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
Manufacturing	604,368,729.43	1,250,743,290.27
Kansas City Plant	604,368,729.43	604,368,729.43
St Louis	125,547,866.13	125,547,866.13
Dallas	139,263,132.99	139,263,132.99
Kansas City	210,270,945.82	210,270,945.82
Seattle	129,286,784.49	129,286,784.49

Example: Using Conditional Sort Options

The following request has a BY HIERARCHY command with a WHEN phrase to select members as well as a WHEN phrase to control the UNDER-LINE option.

The SUBFOOT option is in an ON phrase that references the same hierarchy field (all of the BY options could have been on the BY HIERARCHY phrase).

Each BY option has its own WHEN phrase. The WHEN phrase for the SUBFOOT option uses a measure field in its expression. The WHEN phrase for the UNDER-LINE option uses a hierarchy field in its expression. Attribute fields are not supported in WHEN phrases. They can only be used in IF or WHERE tests.

In this example, one BY option is activated WHEN the MANUFACTURING_CAPTION contains *City*. The other BY option creates a subfoot WHEN SALES_DOLLARS is greater than or equal to $600,000,000:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'Plant' ;
SHOW TOP UNDER-LINE WHEN MANUFACTURING_CAPTION CONTAINS 'City';
ON MANUFACTURING_CAPTION SUBFOOT
" "
"The Sum is large"
" "
WHEN SALES_DOLLARS GE 600000000;
ON TABLE SUBHEAD
"Using BY Options With WHEN on a Measure and a Hierarchy Field"
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using BY Options With WHEN on a Measure and a Hierarchy Field

MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
Manufacturing	1,250,743,290.27	1,250,743,290.27
The Sum is large		
Denver Plant	392,716,985.27	392,716,985.27
Denver	129,327,743.98	129,327,743.98
Salem	130,976,032.31	130,976,032.31
San Diego	132,413,208.98	132,413,208.98
Kansas City Plant	604,368,729.43	604,368,729.43
The Sum is large		
St Louis	125,547,866.13	125,547,866.13
Dallas	139,263,132.99	139,263,132.99
Kansas City	210,270,945.82	210,270,945.82
Seattle	129,286,784.49	129,286,784.49
Columbia Plant	253,657,575.57	253,657,575.57
Bangor	131,243,866.46	131,243,866.46
Columbia	122,413,709.11	122,413,709.11

Example: Using Two BY HIERARCHY Phrases

The following request has a BY HIERARCHY phrase for the Products2 hierarchy and a second BY HIERARCHY phrase for the Manufacturing2 hierarchy (which is from a different dimension, as required). All selected members for the Manufacturing2 hierarchy are repeated for each selected member of the Products2 hierarchy. The WHERE TOTAL phrase omits rows in which SALES_DOLLARS is less than $1,000,000:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'PDA';
BY MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'City';
WHERE TOTAL SALES_DOLLARS GE 1000000;
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using Two BY HIERARCHY Phrases"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using Two BY HIERARCHY Phrases

PRODUCTS_CAPTION	MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
PDA	Kansas City Plant	4,378,416.60	4,378,416.60
	Dallas	1,405,923.22	1,405,923.22
	Seattle	1,084,629.34	1,084,629.34
AMD MICRO	Kansas City Plant	4,378,416.60	4,378,416.60
	Dallas	1,405,923.22	1,405,923.22
	Seattle	1,084,629.34	1,084,629.34
AMD DESKTOP	Kansas City Plant	4,378,416.60	4,378,416.60
	Dallas	1,405,923.22	1,405,923.22
	Seattle	1,084,629.34	1,084,629.34

Example: Using BY HIERARCHY and BY Phrases

The following request has a BY phrase for the Manufacturing2 hierarchy and a BY HIERARCHY phrase for the Products2 hierarchy. All selected members for the Products2 hierarchy are repeated for each selected member of the Manufacturing2 hierarchy (which is not displayed with hierarchical indentations when referenced in a BY phrase). A BY on a unique field is required. Therefore, there is one BY on a unique field with the NOPRINT option and another BY on the caption field:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_MEMBER NOPRINT
BY MANUFACTURING_CAPTION
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'PDA';
WHERE MANUFACTURING_CAPTION CONTAINS 'Plant';
WHERE TOTAL SALES_DOLLARS GE 1000000;
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using BY HIERARCHY and BY Phrases"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using BY HIERARCHY and BY Phrases

MANUFACTURING_CAPTION	PRODUCTS_CAPTION	Sales Dollars	FSUM Sales Dollars
Denver Plant	PDA	3,209,788.86	3,209,788.86
	AMD MICRO	3,209,788.86	3,209,788.86
	AMD DESKTOP	3,209,788.86	3,209,788.86
Kansas City Plant	PDA	4,378,416.60	4,378,416.60
	AMD MICRO	4,378,416.60	4,378,416.60
	AMD DESKTOP	4,378,416.60	4,378,416.60
Columbia Plant	PDA	2,028,561.36	2,028,561.36
	AMD MICRO	2,028,561.36	2,028,561.36
	AMD DESKTOP	2,028,561.36	2,028,561.36

Example: Using ON HIERARCHY Without BY HIERARCHY

The following request has a BY phrase and an ON HIERARCHY phrase for the Manufacturing2 hierarchy. All hierarchy options and BY options are supported on the ON HIERARCHY phrase. This request also has a WHERE clause that selects rows based on the value of a hierarchy field (MANUFACTURING_CAPTION). The WHERE test selects rows based on values at the leaf nodes:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS FROLL.SALES_DOLLARS
BY MANUFACTURING_CAPTION
ON MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'Plant';
WHERE MANUFACTURING_CAPTION CONTAINS 'City';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using ON HIERARCHY"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using ON HIERARCHY

MANUFACTURING_CAPTION	Sales Dollars	FSUM Sales Dollars
Manufacturing	.	.
Kansas City Plant	210,270,945.82	210,270,945.82
Kansas City	210,270,945.82	210,270,945.82

Reference: **Hierarchical Reports With Multiple Display Commands**

In a request with more than one display command, each command must repeat all of the sort phrases from the previous command in the same order after which additional sort phrases can be added. In a hierarchical reporting request, the following rules must be followed:

- All display commands must be summation commands. PRINT is not supported in a hierarchical reporting request.
- Both BY and BY HIERARCHY sort phrases can be used. As with a non-hierarchical request, all of the sort phrases must be repeated in the same order with a subsequent display command.
- Only one WHEN and one SHOW phrase can be specified for each hierarchy referenced in the request. These phrases can be specified at any level of the request, but they will apply to every display command in the request.

Example: Using Multiple Display Commands in a Hierarchical Report

The following request has two WRITE commands. The first WRITE command has a BY HIERARCHY sort phrase for the Products2 hierarchy and a BY HIERARCHY phrase for the Manufacturing2 hierarchy. The second WRITE command repeats these phrases and adds a BY phrase for the Customers2 hierarchy:

```
TABLE FILE CENTSALE
WRITE SALES_DOLLARS
BY PRODUCTS_CAPTION HIERARCHY
WHEN PRODUCTS_CAPTION EQ 'PDA';
SHOW TO DOWN 0
BY MANUFACTURING_CAPTION HIERARCHY
WHEN MANUFACTURING_CAPTION CONTAINS 'Plant';
SHOW TO DOWN 0
WRITE SALES_DOLLARS UNITS
BY PRODUCTS_CAPTION HIERARCHY
BY MANUFACTURING_CAPTION HIERARCHY
BY CUSTOMERS_CAPTION
WHERE CUSTOMERS_MEMBER LIKE '01%'
WHERE TOTAL SALES_DOLLARS NE 0

ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using Multiple Display Commands"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The BY HIERARCHY phrase for the Products2 hierarchy has a WHEN phrase that selects members whose caption is *PDA*. The SHOW option (TO DOWN 2) displays two levels of descendants below the PDA level.

For the Manufacturing2 hierarchy, the selected members have the characters *Plant* in their captions. The SHOW option displays no descendants on the report output. All selected members of the Manufacturing2 hierarchy display for each selected member of the Products2 hierarchy.

The WHERE TOTAL test removes all rows with zero SALES_DOLLARS from the report output. The WHERE test selects customers whose member values begin with 01.

The output is:

Using Multiple Display Commands

PRODUCTS_CAPTION	MANUFACTURING_CAPTION	Sales Dollars	CUSTOMERS_CAPTION	Sales Dollars	Units
PDA	Denver Plant	11,549.78	.	11,549.78	11.00
	Kansas City Plant	9,449.82	.	9,449.82	9.00
	Columbia Plant	3,149.94	.	3,149.94	3.00
AMD MICRO	Denver Plant	11,549.78	.	11,549.78	11.00
	Kansas City Plant	9,449.82	.	9,449.82	9.00
	Columbia Plant	3,149.94	.	3,149.94	3.00
AMD DESKTOP	Denver Plant	11,549.78	ACE`S, INC.	9,449.82	9.00
			ANCONA BROTHERS	2,099.96	2.00
	Kansas City Plant	9,449.82	VSA INC	9,449.82	9.00
	Columbia Plant	3,149.94	SUNSET DISTRIBUTING	3,149.94	3.00

Columnar Reporting

If you run a standard TABLE or FML report against a cube, you are running what is known as a columnar report. A columnar report flattens out the multi-dimensional structure of a cube to show information from one angle.

To create a valid columnar report, you must understand the data structure of the cube and the logic and assumptions that the adapter uses when retrieving values from a cube.

The adapter has built-in logic that enables it to determine the rows and cells to extract from rollups in order to deliver reports. TABLE uses the lowest level BY phrase for each dimension to determine at which rollup to locate the lookup for aggregations. It then rolls the result up from there, performing aggregations on the data as needed to ensure the consistency of the results.

If you do not specify a BY field in your TABLE request, you may receive invalid data or a message. This happens because without a BY field to determine the level of rollup at which to perform the data lookup, TABLE does not know where to source the result. At least one BY phrase (perhaps with the NOPRINT option) must be on a field with unique values.

Reference: Reporting on Measures

The Essbase Server does not support the use of SUM or PRINT on an Measure without reference to another dimension in the request.

For example, since this request refers only to measure,

```
TABLE FILE BASIC
PRINT DATA_VALUE
END
```

the following message displays:

```
(FOC43244) No active dimension found in the request.
```

Reference: Columnar Reporting on a Parent/Child View

You cannot reference PARENT or PARENTCAP from the parent/child view without referencing MEMBER or CAPTION, unless you are using it as a filter.

Example: Referencing MEMBER and CAPTION Attributes

The following request is incorrect since it does not reference MEMBER or CAPTION.

```
TABLE FILE BASIC
PRINT DATA_VALUE
BY PRODUCT_PARENT
WHERE PRODUCT_PARENT EQ 'Visual'
END
```

The following message displays:

```
(FOC43248) Can't reference PRODUCT's PARENT,PARENTCAP without MEMBER or
CAPTION.
```

The correct request is:

```
TABLE FILE BASIC
PRINT DATA_VALUE
BY PRODUCT_PARENT
BY PRODUCT_MEMBER
WHERE PRODUCT_PARENT EQ 'Visual'
END
```

The output is:

```
PRODUCT_PARENT    PRODUCT_MEMBER        DATA_VALUE
--------------    --------------        ----------
Visual            Camera                 20,971.00
                  Television             27,877.00
                  VCR                    49,206.00
```

You cannot reference fields from the Generation view (GEN1_PRODUCT or GEN2_PRODUCT) and from the parent/child view (PRODUCT_MEMBER or PRODUCT_CAPTION) in one request if the fields from these different views are from the same dimension.

For example, in the following request

```
TABLE FILE BASIC
PRINT DATA_VALUE
BY PRODUCT_MEMBER
BY FAMILY
WHERE PRODUCT_MEMBER EQ 'Visual'
END
```

FAMILY is a member of the Product dimension and part of the Generation view so the following message is generated:

```
(FOC43247) Can't reference fields from Product2 and Product at the same
time.
```

The correct request is:

```
TABLE FILE BASIC
PRINT DATA_VALUE
BY PRODUCT_MEMBER
BY SCENARIO
WHERE PRODUCT_MEMBER EQ 'Visual'
END
```

The output is:

```
PRODUCT_MEMBER  SCENARIO     DATA_VALUE
--------------  --------     ----------
Camera          Scenario      20,971.00
Television      Scenario      27,877.00
VCR             Scenario      49,206.00
```

Syntax:

How to Collapse PRINT With ACROSS

The PRINT command generates a report that has a single line for each record retrieved from the data source after screening out those that fail IF or WHERE tests. When PRINT is used in conjunction with an ACROSS phrase, many of the generated columns may be empty. Those columns display the missing data symbol.

To avoid printing such a sparse report, you can use the SET ACROSSPRT command to compress the lines in the report. The number of lines is reduced within each sort group by swapping non-missing values from lower lines with missing values from higher lines, and then eliminating any lines whose columns all have missing values.

Because data may be moved to different report lines, row-based calculations such as ROW-TOTAL and ACROSS-TOTAL in a compressed report are different from those in a non-compressed report. Column calculations are not affected by compressing the report lines.

The syntax is

```
SET ACROSSPRT = {NORMAL|COMPRESS}
```

```
ON TABLE SET ACROSSPRT {NORMAL|COMPRESS}
```

where:

NORMAL

Does not compress report lines.

COMPRESS

Compresses report lines by promoting data values up within a sort group.

Example: Compressing Report Output

The following request against the Adventure Works cube prints average sales by country across calendar year:

```
TABLE FILE ADVENTURE_WORKS
PRINT INTERNET_AVERAGE_SALES_AMOUNT/D8.1 AS 'Average Sales'
BY COUNTRY51
ACROSS CALENDAR_YEAR4
WHERE COUNTRY51 EQ 'France' OR 'Germany' OR 'United Kingdom'
ON TABLE SET ACROSSPRT NORMAL
ON TABLE SET PAGE-NUM OFF
END
```

Each line of the report output represents one value of average sales for one year, so at most one column on each line has a value when ACROSSPRT is set to NORMAL (the default). The output is:

	Calendar Year			
	CY 2001	CY 2002	CY 2003	CY 2004
Country	Average Sales	Average Sales	Average Sales	Average Sales
France	3,084.2	.	.	.
	.	2,232.1	.	.
	.	.	1,000.7	.
	.	.	.	802.7
Germany	3,133.9	.	.	.
	.	2,275.9	.	.
	.	.	1,081.3	.
	.	.	.	909.6
United Kingdom	3,020.0	.	.	.
	.	2,261.5	.	.
	.	.	1,047.5	.
	.	.	.	856.5

Running the same request with ACROSSPRT=COMPRESS moves data values from lower lines to higher lines within the same BY group and eliminates lines that have all missing values:

```
TABLE FILE ADVENTURE_WORKS
PRINT INTERNET_AVERAGE_SALES_AMOUNT/D8.1 AS 'Average Sales'
BY COUNTRY51
ACROSS CALENDAR_YEAR4
WHERE COUNTRY51 EQ 'France' OR 'Germany' OR 'United Kingdom'
ON TABLE SET ACROSSPRT COMPRESS
ON TABLE SET PAGE-NUM OFF
END
```

The output is:

	Calendar Year			
	CY 2001	CY 2002	CY 2003	CY 2004
Country	Average Sales	Average Sales	Average Sales	Average Sales
France	3,084.2	2,232.1	1,000.7	802.7
Germany	3,133.9	2,275.9	1,081.3	909.6
United Kingdom	3,020.0	2,261.5	1,047.5	856.5

Full and Partial Aggregation

Cube requests are requests that reference at least one measure and possibly more than one hierarchy. Attributes may also be referenced.

For cube requests, the adapter reads cells with fully or partially aggregated data from the cube and applies certain restrictions on the contents of the TABLE request to ensure consistency of reports.

There are two modes in which cube requests are processed: *Full Aggregation* and *Partial Aggregation*.

- In full aggregation mode, the adapter retrieves aggregated values from the cube exactly as they will be displayed on the report.
- In partial aggregation mode the Adapter retrieves partially aggregated data which is then further aggregated by WebFOCUS.

Full aggregation mode is more efficient because no aggregation is needed beyond what is already stored in the cube.

Reference: Support for Full Aggregation Mode

The following table describes support for full aggregation mode:

Command	Dimension Properties	Measures	Attributes
WRITE/ SUM/ADD	Supported, converted to FST.	Supported, converted to an operation that matches the measure aggregator.	Not supported.

Command	Dimension Properties	Measures	Attributes
COMPUTE	Supported.	Supported.	Not supported.
PRINT	Supported.	Supported for compatibility. Not supported with BY HIERARCHY.	Not supported.
WHERE TOTAL	Supported	Supported	Not supported.
WHERE	Supported	Supported with RESTRICTSUM.	Required for attributes. Must provide required restriction for its variable selection type. Restricted to a group of fields that represent a single attribute.
WHEN	Supported	Not supported.	Not supported.
DEFINE	Supported	Supported with RESTRICTSUM.	Not supported.
BY	Supported. Requires BY on a unique field of a parent/ child hierarchy (except when referenced only in a WHERE). BY HIERARCHY adds BY internally.	Not supported.	Not supported.
BY HIERARCHY	Supported.	Not supported.	Not supported.

Command	Dimension Properties	Measures	Attributes
ON HIERARCHY	Supported. Requires a corresponding BY or BY HIERARCHY phrase.	Not supported.	Not supported.

Reference: Support for Partial Aggregation Mode

In partial aggregation mode, there are additional restrictions on the use of measures. Note that in requests with multiple display commands, the partial aggregation restrictions apply to all display commands except for the ones associated with the lowest BY phrase:

- Additive aggregators (those that can be further aggregated from partial totals such as SUM., CNT., MAX., and MIN.) are supported as long as they match the measure aggregator.
- Non-additive aggregators (such as AVE.) are not supported.

The following factors turn on partial aggregation mode:

- Multiple display commands.
- Parent/Child hierarchies: Visual totals or Full totals with a WHERE on the hierarchy.
- Level hierarchies: not all levels of a hierarchy that are referenced in the request have BY phrases.
- All hierarchies: the hierarchy is referenced only in a WHERE clause that selects more than one member.

Reporting Rules

The reporting rules and features vary depending on the types of hierarchies defined in the synonym. These factors are summarized in the following sections.

Reference: Display Command and Prefix Operator Support

When BY HIERARCHY is not used in the request, the display command can be PRINT, even for aggregated data. It is internally converted to the appropriate command.

Note: To emphasize that the summary values are already in the cube and do not necessarily require any additional aggregation, the examples use the WRITE command, which is a synonym for SUM.

The measure operation is also automatically supplied internally; you do not have to specify a prefix operator in the request. However, if you do specify a prefix operator, it should match the measure aggregator.

Prefix operators in a request have an effect in two situations. You can use the FROLL prefix operator to specify a full total in a BY HIERARCHY request. Also, requests with multiple display commands (and sort groups) use the prefix operator to determine the sort order of the high level sort groups.

Reference: Reporting Rules for All Hierarchies

BY. Lexicographic sort is available on any of the member properties as well as expressions (DEFINEs) based on properties, but the request must include a sort (possibly with the NOPRINT option) on a property that is guaranteed to uniquely identify a hierarchy member, either PROPERTY=UID or PROPERTY=NAME. BY HIERARCHY automatically adds a BY one of these properties. In a report referencing a level hierarchy, the request must specify one.

Hierarchies. Only one hierarchy from the same dimension can be represented on the report.

Measures. Filtering on totals is done using WHERE or WHERE TOTAL tests. Calculations on totals are done using DEFINE or COMPUTE (RESTRICTSUM ON).

Screening. Screening on member properties is available with and without sorting.

Joins. The adapter does not support direct JOINs from or to a cube. Therefore, SQL joins using a WHERE clause are not supported. To achieve a joined request, use the WebFOCUS MATCH FILE command or create a HOLD file and use the HOLD file in the request.

Reference: Reporting Rules for Parent/Child Hierarchies

BY HIERARCHY. Requires the command SUM, WRITE, or ADD. PRINT is not supported.

Hierarchical Context. The set of members selected for the report using a WHEN test is augmented so that every member of the hierarchy is shown within its hierarchical context. The displayed subtree of hierarchy members is a contiguous tree whose root is the root of the whole hierarchy. The added context members are shown without totals.

Report Compression. By default, hierarchy members with no fact data do not participate in the report. Hierarchy members for which no data passes screening do not participate in the report. In a multi-hierarchy report, the inner hierarchies are not shown for those members of outer hierarchies that appear on the report output only to show the context of selected members.

Full and visual totals. Visual totals for each node are composed from the totals of its hierarchical children shown on the same report. These are compatible with the totals shown for the same members in a multi-verb level-wise report. Visual totals are the default. The report can specify full totals by specifying the prefix operator FROLL on a measure field. Full totals are accumulated for each displayed hierarchy member individually. These are compatible with the totals shown for the same members in a non-hierarchical report on a parent-child hierarchy.

General Tips for Reporting

When creating a report request:

- PRINT and SUM can be used interchangeably when using level hierarchies.
- To maximize reporting efficiency, always sort in the same direction as the hierarchy for the cube. For example, if the hierarchy for a dimension is

  Country > State_Province > City

  sorting by City first, then State_Province results in an inefficient request. Also, because of the hierarchical structure of OLAP, you retrieve only one BY field for each SUM or PRINT field in the report.

Chapter 23

Using the Adapter for EXASol

The Adapter for EXASol allows applications to access EXASolution data sources. The adapter converts data or application requests into native EXASol SQL statements and returns optimized answer sets to the requesting program.

In this chapter:

Introducing the Adapter for EXASol

The Adapter for EXASol is ODBC based. It utilizes EXASolution ODBC driver version 5.0.15 or higher, that can be installed on Windows and Linux.

Preparing the EXASol ODBC Environment

On Windows, the EXASol ODBC environment is set up during the installation of the EXASolution ODBC driver. No additional setup steps are required.

On Linux, after the EXASolution ODBC driver is installed, make sure that path to the directory where EXASol driver installed is included in LD_LIBRARY_PATH (or in IBI_LIBPATH if the server is running with security on).

For example:

```
export IBI_LIBPATH=/rdbms/exasol/EXASolution_ODBC-5.0.15/lib/x86_64
```

Configuring the Adapter for EXASol

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the EXASol Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. Right-click *EXASol* and click *Configure*.
4. Select *edasprof* from the drop-down menu to add this connection for all users, or select a user profile.
5. Click *Test*. You should see a sample list of objects stored in the EXASol data source you are accessing.
6. Click *Configure*.

Reference: EXASol Adapter Configuration Settings

The Adapter for EXASol is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

System

Is the name of the machine where the EXASol in-memory database is running. The basic syntax is *host*:*port*

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Additional connection string keywords (optional)
Can be used to add options to the connection string, for example, EXASCHEMA.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Reference: Configuring an Auxiliary Connection for Extended Bulk Load

Using Extended Bulk Load with the EXASol adapter requires that intermediate data files are transferred to a server where the EXASol sever can access them.

For this adapter, you must enable the Flat File adapter and create a connection using FTP or SFTP to the server. From the DMC or Web Console, perform the following steps:

1. Expand the *Adapters* folder or click the *Adapters* tab.
2. Right-click *Flat File* and click *Add Connection*.
3. In the Connection Name box, type the name of the connection for your EXASol server. The connection names must match.
4. From the Protocol drop-down menu, select *FTP* or *SFTP*.
5. In the Host Name box, type the name of the host for your server.

   Note: For EXASol it should not be the system where it is running, rather it should be some other system that it can access using FTP.
6. In the Initial Directory box, type the name of a directory that you have write access to, such as your home directory.

7. In the User and Password boxes, type your user ID and password.
8. Click *Test*.

   The following message should appear: *User has read access to the FTP server*.
9. Click *OK*.
10. Click *Configure*.
11. Click *Close*.

Creating Synonyms With EXASol

Synonyms define unique names, or aliases, for each EXASol table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given EXASol table.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Using Direct Pass-through With EXASol

Direct Pass-through commands, such as SHOW TABLES and DESCRIBE tablename, do not display any results.

This can be avoided by adding a comment:

```
ENGINE SQLEXA
/*QUERY*/ SHOW TABLES ;
END
```

Chapter 24

Using the Adapter for Excel

The Adapter for Excel allows applications to access Excel data sources. The adapter returns answer sets to the requesting application.

In this chapter:

Configuring the Adapter for Excel

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to the Excel database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Excel database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Excel Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Excel

The Excel adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

This must be the name of an ODBC data source configured for the "Microsoft Excel Driver." If a File DSN is used, this must be the name of the file (for example, samplefdsn.dsn).

Security

There are two methods by which a user can be authenticated when connecting to a data source:

- **Explicit.** Not applicable for Excel. Reserved for future use.
- **Trusted.** The adapter connects to the EXCEL data source as a Windows login using the credentials of the operating system user imitated by the server data access agent (Default).

User

Not applicable for Excel. Reserved for future use.

Password

Not applicable for Excel. Reserved for future use.

File DSN parameters (optional)

Is the path to the ODBC File DSN that is defined on your Windows machine or network.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

For User/System DSN:

```
ENGINE SQLEXCEL SET CONNECTION_ATTRIBUTES connection DSN_name/,
```

For File DSN:

```
ENGINE [SQLEXCEL] SET CONNECTION_ATTRIBUTES connection
 FileDSN_name/,:'filedsn=FileDSN_path;',
```

where:

`SQLEXCEL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_name`

Is the Excel Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`FileDSN_name`

Is the name of Excel File Data Source Name (DSN) you wish to access.

`FileDSN_path`

Is the path to the Excel File DSN, which may reside on the hard drive or on the network.

Example: Declaring Connection Attributes

For User/System DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the Excel DSN named SAMPLESERVER.

```
ENGINE SQLEXCEL SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/,
```

For File DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 using the ODBC File DSN named SAMPLESERVER.dsn.

```
ENGINE SQLEXCEL SET CONNECTION_ATTRIBUTES CON2
SAMPLESERVER.dsn /,:'filedsn=C:\ SAMPLESERVER.dsn;'
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLEXCEL SET DEFAULT_CONNECTION connection
```

where:

`SQLEXCEL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLEXCEL SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLEXCEL SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLEXCEL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Excel Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Excel data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLEXCEL to identify the Adapter for Excel.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLEXCEL [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLEXCEL

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each Excel table or sheet that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Excel

The following list describes the synonym creation parameters for which you can supply values.

Workbook

Select a Workbook. The default is the workbook selected for the ODBC source.

You can type in a different workbook name or click on the ellipsis (...). This opens the Select Excel Workbook dialogue with the workbook name entered selected.

Filter by Object name

Selecting this option adds the Object Name parameters to the screen.

Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

The list of tables consists of:

- Range names in the workbook.

- Sheet names. These names have a $ at the end of the name.

Example: **Sample Generated Synonym**

An Adapter for Excel synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File

```
FILENAME=ONHAND, SUFFIX=SQLEXCEL, $
  SEGMENT=ONHAND, SEGTYPE=S0, $
    FIELDNAME=PROD_NUM, ALIAS=Prod_Num, USAGE=A255V, ACTUAL=A255V,
      MISSING=ON, $
    FIELDNAME=PRODNAME, ALIAS=Prodname, USAGE=A255V, ACTUAL=A255V,
      MISSING=ON, $
    FIELDNAME=QTY_IN_STOCK, ALIAS=Qty_in_Stock, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=PRICE, ALIAS=Price, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=COST, ALIAS=Cost, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
```

Access File

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,WRITE=YES,$
```

Reference: **Access File Keywords**

This chart describes the keywords in the Access File for Excel.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Excel table. The value assigned to this attribute can include the name of the workbook as follows: TABLENAME=[*filename*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96

Customizing the Excel Environment

The Adapter for Excel provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Excel.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLEXCEL SET TIMEOUT {nn|0}
```

where:

`SQLEXCEL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLEXCEL SET PASSRECS {ON|OFF}
```

where:

`SQLEXCEL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Excel Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 25 Using the Adapter for Excel (via Direct Retrieval)

The Adapter for Excel (via direct retrieval) supports direct retrieval from Excel files (.xls and .xlsx files). The adapter uses the Apache POI, which is a Java API that provides access to Microsoft file formats.

In this chapter:

Configuring the Adapter for Excel (via Direct Retrieval)

You can configure the Adapter for Excel (via direct retrieval) from the Web Console or the Data Management Console.

Procedure: How to Configure the Adapter for Excel (via Direct Retrieval)

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. Right-click Excel (via direct retrieval), and select *Configure*.

   No input parameters are required.

   The configured adapter is added to the Adapters list in the navigation pane.

Managing Metadata for Excel (via Direct Retrieval)

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Excel worksheet that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server metadata.

Note that certain settings can affect synonym creation. For more information on the adapter settings, see *Changing Adapter Settings* on page 719.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Applications or Adapters pages of the Web Console.

1. From the Web Console sidebar, click *Connect to Data*.

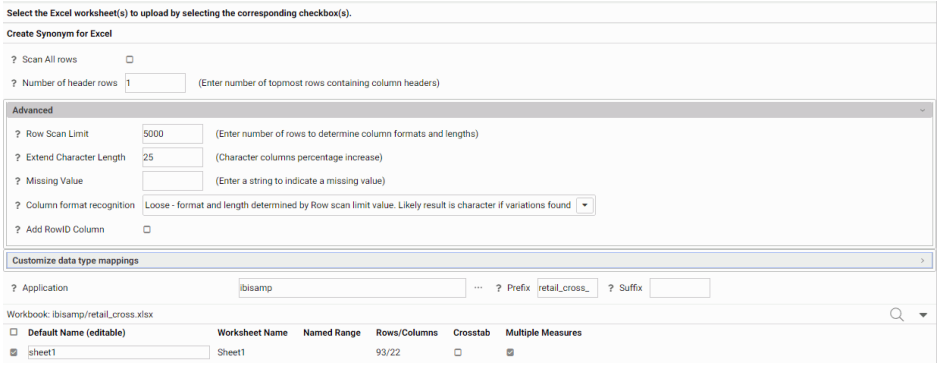
   The Adapters page opens.

2. Right-click a connection and click *Show Files*.

   A file picker dialog box opens.

3. Navigate to the application and Excel file for which you want to create the synonym and Click OK.

The *Select Excel Worksheets* page opens, as shown in the following image.



4. Enter or select values for the following parameters.

`Scan All Rows`
: If checked, scans all rows to determine data types and lengths.

`Row Scan Limit`
: Specifies the maximum number of rows to scan to determine data types and lengths. The default value is 5000.

`Extend Character Length`
: Is a percentage to add to the characters lengths to compensate for a partial scan. The default value is 25.

`Missing Value`
: Enter a string to indicate a missing value.

`Column format recognition`
: Can be Loose, to use the row scan limit to determine types and lengths, or Strict, to determine data type using the first row and the lengths using the row scan limit. The default value is Loose.

`Number of header rows`
: Enter the number of header rows containing column names. The default value is 1.

`Add RowID Column`
: If checked, adds a row number for each spreadsheet row.

`Customize data type mappings`
: If expanded, opens additional fields for decomposing date fields, adding geographic roles, and selecting the data type for numeric fields and for alphanumeric fields.

`Application`
: Enter an application name or click the ellipsis (...) to navigate to the application for the synonym.

`Prefix`

Assign a prefix to the synonym names.

`Suffix`

Assign a suffix to the synonym names.

`Workbook`

Select the workbooks for which synonyms are to be created. You can change the synonym name by selecting the default name and typing a new name.

`Crosstab`

When selected, the header rows, up to the number specified, are un-pivoted to become data values.

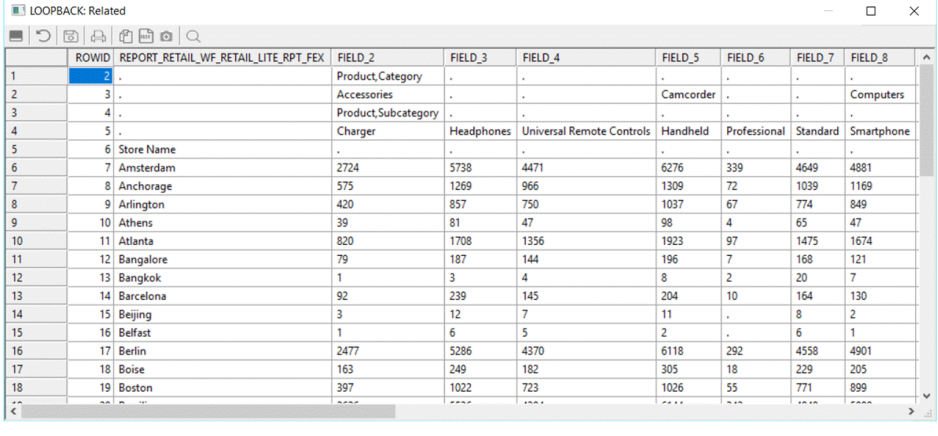
`Multiple Measures`

Check *Multiple Measures* if there is more than one numeric value for the titles in the last header row.

5. Click the *Create Base Synonyms* button on the ribbon.

Example: Creating a Crosstabbed Synonym

The following shows a partial Excel Spreadsheet open in the Data Management Console. The headers include Product Category, a value for Product Category, Product Subcategory, and a value for Product Subcategory.



	ROWID	REPORT_RETAIL_WF_RETAIL_LITE_RPT_FEX	FIELD_2	FIELD_3	FIELD_4	FIELD_5	FIELD_6	FIELD_7	FIELD_8
1	2	.	Product,Category	.	.	.	.	.	.
2	3	.	Accessories	.	.	Camcorder	.	.	Computers
3	4	.	Product,Subcategory	.	.	.	.	.	.
4	5	.	Charger	Headphones	Universal Remote Controls	Handheld	Professional	Standard	Smartphone
5	6	Store Name	.	.	.	.	.	.	.
6	7	Amsterdam	2724	5738	4471	6276	339	4649	4881
7	8	Anchorage	575	1269	966	1309	72	1039	1169
8	9	Arlington	420	857	750	1037	67	774	849
9	10	Athens	39	81	47	98	4	65	47
10	11	Atlanta	820	1708	1356	1923	97	1475	1674
11	12	Bangalore	79	187	144	196	7	168	121
12	13	Bangkok	1	3	4	8	2	20	7
13	14	Barcelona	92	239	145	204	10	164	130
14	15	Beijing	3	12	7	11	.	8	2
15	16	Belfast	1	6	5	2	.	6	1
16	17	Berlin	2477	5286	4370	6118	292	4558	4901
17	18	Boise	163	249	182	305	18	229	205
18	19	Boston	397	1022	723	1026	55	771	899

With Crosstab not checked, the resulting Master File has a field for column on the Spreadsheet. The following is a partial list of fields for the Spreadsheet:

```
FIELDNAME=PRODUCT_CATEGORY_ACCESSORIES_PRODUCT_SUBCATEGORY,
ALIAS='Product,Category_Accessories_Product,Subcategory', USAGE=A8V,
ACTUAL=A8V,
      MISSING=ON,
      TITLE='Product,Category_Accessories_Product,Subcategory', $
    FIELDNAME=FIELD_3, ALIAS=FIELD_3, USAGE=A12V, ACTUAL=A12V,
      MISSING=ON,
      TITLE='FIELD_3', $
    FIELDNAME=FIELD_4, ALIAS=FIELD_4, USAGE=A31V, ACTUAL=A31V,
      MISSING=ON,
      TITLE='FIELD_4', $
    FIELDNAME=CAMCORDER, ALIAS=Camcorder, USAGE=A10V, ACTUAL=A10V,
      MISSING=ON,
      TITLE='Camcorder', $
    FIELDNAME=FIELD_6, ALIAS=FIELD_6, USAGE=A15V, ACTUAL=A15V,
      MISSING=ON,
      TITLE='FIELD_6', $
    FIELDNAME=FIELD_7, ALIAS=FIELD_7, USAGE=A10V, ACTUAL=A10V,
      MISSING=ON,
      TITLE='FIELD_7', $
    FIELDNAME=COMPUTERS, ALIAS=Computers, USAGE=A12V, ACTUAL=A12V,
      MISSING=ON,
      TITLE='Computers', $
    FIELDNAME=FIELD_9, ALIAS=FIELD_9, USAGE=A7V, ACTUAL=A7V,
      MISSING=ON,
      TITLE='FIELD_9', $
    FIELDNAME=MEDIA_PLAYER, ALIAS='Media Player', USAGE=A8V, ACTUAL=A8V,
      MISSING=ON,
      TITLE='Media Player', $
```

With Crosstab checked, multiple measures unchecked, and five header rows specified, PRODUCT_CATEGORY becomes one dimension field, PRODUCT_SUBCATEGORY becomes a second dimension field, and the numeric values in the Spreadsheet become a measure field. The first column (containing store names) becomes an alphanumeric field. For example:

LOOPBACK: Data for foccache/cross_demo_sheet1. Limit 500 rows.

Double the Row Limit

	Store Name	Product Category	Product Subcategory	Measure
1	Amsterdam	Accessories	Charger	2724
2	Amsterdam	Accessories	Headphones	5738
3	Amsterdam	Accessories	Universal Remote Controls	4471
4	Amsterdam	Camcorder	Handheld	6276
5	Amsterdam	Camcorder	Professional	339
6	Amsterdam	Camcorder	Standard	4649
7	Amsterdam	Computers	Smartphone	4881
8	Amsterdam	Computers	Tablet	2841
9	Amsterdam	Media Player	Blu Ray	17439
10	Amsterdam	Media Player	DVD Players	1232
11	Amsterdam	Media Player	DVD Players - Portable	361
12	Amsterdam	Media Player	Streaming	1591
13	Amsterdam	Stereo Systems	Boom Box	619
14	Amsterdam	Stereo Systems	Home Theater Systems	10059

The resulting Master File follows:

```
FILENAME=RETAIL_CROSS_CROSS_ON, SUFFIX=DIREXCEL,
 DATASET=ibisamp/retail_cross.xlsx, BV_NAMESPACE=OFF, $
  SEGMENT=RETAIL_CROSS_CROSS_ON, SEGTYPE=S0, $
    FIELDNAME=DHL1, ALIAS=DHL1, USAGE=A20V, ACTUAL=A20V,
      MISSING=ON,
      TITLE='DHL1', $
    FIELDNAME=DVL1_PRODUCT_CATEGORY, ALIAS='Product,Category',
      USAGE=A20V, ACTUAL=A20V,
      MISSING=ON,
      TITLE='Product,Category', $
    FIELDNAME=DVL2_PRODUCT_SUBCATEGORY, ALIAS='Product,Subcategory',
      USAGE=A31V, ACTUAL=A31V,
      MISSING=ON,
      TITLE='Product,Subcategory', $
    FIELDNAME=REPORT_RETAIL_WF_RETAIL_LITE_RPT_FEX,
      ALIAS='Report retail/wf_retail_lite_rpt.fex',
      USAGE=I9, ACTUAL=A11V,
      MISSING=ON,
      TITLE='Report retail/wf_retail_lite_rpt.fex', $
  FOLDER=MG_RETAIL_CROSS_CROSS_ON, PARENT=.,
    DV_ROLE=MEASURE,
    DESCRIPTION='Measure Groups', $
  FOLDER=RETAIL_CROSS_CROSS_ON, PARENT=MG_RETAIL_CROSS_CROSS_ON,
    DESCRIPTION='Retail_cross_cross_on', $
    FIELDNAME=REPORT_RETAIL_WF_RETAIL_LITE_RPT_FEX,
      BELONGS_TO_SEGMENT=RETAIL_CROSS_CROSS_ON, $
  FOLDER=DIM_RETAIL_CROSS_CROSS_ON, PARENT=.,
    DV_ROLE=DIMENSION,
    DESCRIPTION='Dimensions', $
  FOLDER=RETAIL_CROSS_CROSS_ON1, PARENT=DIM_RETAIL_CROSS_CROSS_ON,
    DESCRIPTION='Retail_cross_cross_on', $
    FIELDNAME=DHL1, BELONGS_TO_SEGMENT=RETAIL_CROSS_CROSS_ON, $
  FOLDER=DVL, PARENT=RETAIL_CROSS_CROSS_ON1,
    DESCRIPTION='DVL', $
    FIELDNAME=DVL1_PRODUCT_CATEGORY,
      BELONGS_TO_SEGMENT=RETAIL_CROSS_CROSS_ON,
      DV_ROLE=LEVEL,  $
    FIELDNAME=DVL2_PRODUCT_SUBCATEGORY,
      BELONGS_TO_SEGMENT=RETAIL_CROSS_CROSS_ON,
      DV_ROLE=LEVEL,  $
```

You can now issue a request similar to the following:

```
TABLE FILE RETAIL_CROSS_CROSS_ON
PRINT REPORT_RETAIL_WF_RETAIL_LITE_RPT_FEX AS Measure
BY DHL1 AS Store
BY DVL1_PRODUCT_CATEGORY AS Category
BY DVL2_PRODUCT_SUBCATEGORY AS Subcategory
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

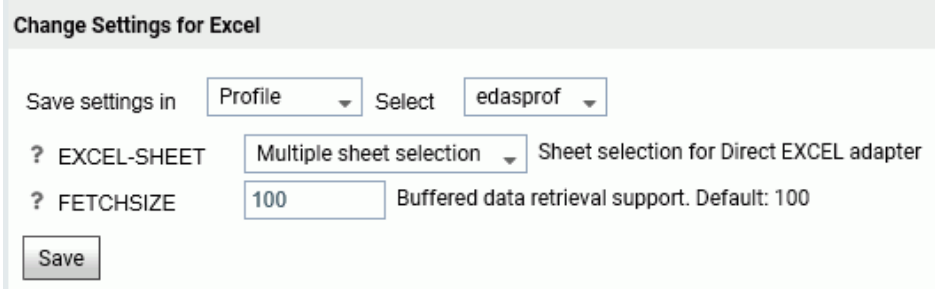
Partial output is shown in the following image.

Store	Category	Subcategory	Measure
Amsterdam	Accessories	Charger	2724
		Headphones	5738
		Universal Remote Controls	4471
	Camcorder	Handheld	6276
		Professional	339
		Standard	4649
	Computers	Smartphone	4881
		Tablet	2841
	Media Player	Blu Ray	17439
		DVD Players	1232
		DVD Players - Portable	361
		Streaming	1591
	Stereo Systems	Boom Box	619
		Home Theater Systems	10059
		Receivers	3833
		Speaker Kits	6257
		iPod Docking Station	7939
	Televisions	CRT TV	333
		Flat Panel TV	2293
		Portable TV	517
	Video Production	Video Editing	4897
Anchorage	Accessories	Charger	575
		Headphones	1269
		Universal Remote Controls	966
	Camcorder	Handheld	1309
		Professional	72
		Standard	1039
	Computers	Smartphone	1169
		Tablet	962
	Media Player	Blu Ray	3588
		Streaming	343
	Stereo Systems	Home Theater Systems	2191
		Receivers	805
		Speaker Kits	1441
		iPod Docking Station	1651
	Televisions	Flat Panel TV	459
	Video Production	Video Editing	1066

Changing Adapter Settings

To change the settings for the adapter, right-click the adapter on the Adapters page and select *Change Settings*.

The *Change Settings for Excel*page opens, as shown in the following image.



You can change the following settings.

EXCEL-SHEET

Select *Multiple sheet selection* (the default) or *Single sheet selection* from the drop-down list. Your selection affects whether you can select multiple worksheets or only a single worksheet when uploading a file using the Upload Wizard feature.

FETCHSIZE

Implements buffered retrieval. Enter a number of rows or accept the default value (100).

Select profile

Select a profile to contain these settings from the drop-down list.

Chapter 26

Using the Adapter for Facebook

This section describes how to configure the Facebook Adapter.

In this chapter:

Facebook Adapter Overview

The Facebook Adapter is used to report against information, resident in the Facebook environment. It can also be used by DataMigrator in the creation of datamarts to house Facebook information.

You can configure the Facebook Adapter using the Reporting Server Web Console. The adapter requires a connection, which stores the access token. A valid Facebook access token is needed to issue Facebook API calls. The token is associated with a Facebook application and a specific Facebook user. The access token is valid for 60 days after which a new access token would need to be obtained and configured.

The connection page shows permissions that will be granted to the Facebook application. You can uncheck permissions that should not be granted. The connection process will use only checked permissions. Authentication will take place even if no permissions are granted.

Creating a Facebook Application

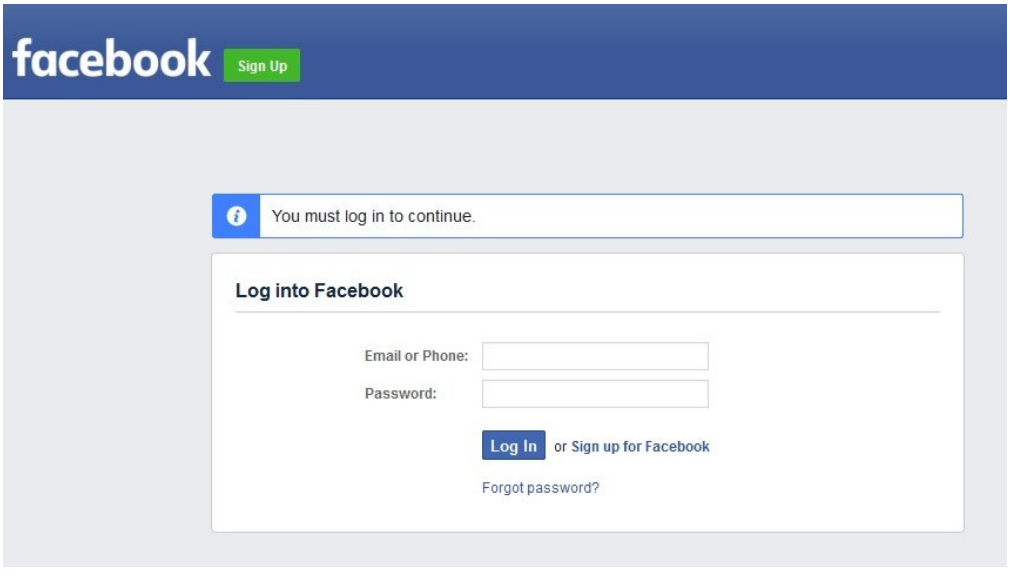
A Facebook application needs to exist before configuring the Facebook Adapter. The Facebook application must be associated with the Domain where the WebFOCUS Reporting Server is installed.

Procedure: How to Create a Facebook Application

1. From a browser, enter the following URL in a web browser:

   `https://developers.facebook.com/apps`

A Log into Facebook screen will appear if you are not already logged into Facebook, as shown in the following image.

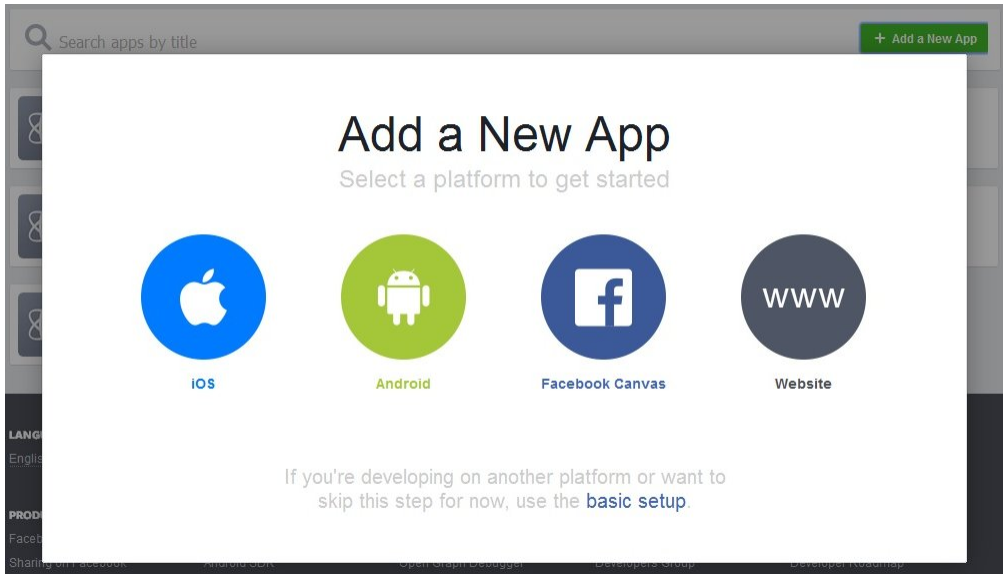


2. Enter your Facebook credentials and click *Log In*.

   The following screen is displayed.

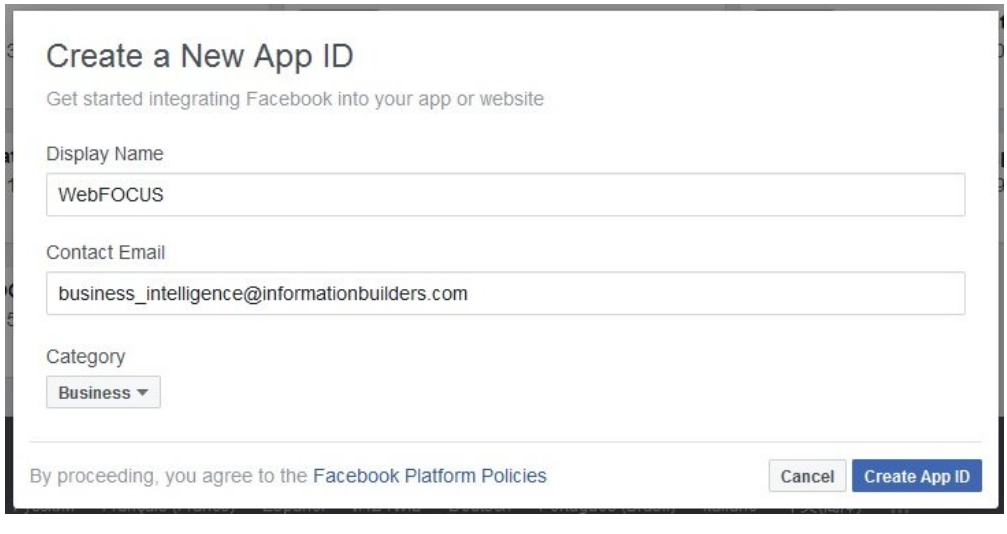
3. Click + *Add a New App*.

The Add a New App screen is displayed, as shown in the following image.



4. Click *basic setup*.

   The Create a New App ID screen is displayed, as shown in the following image



5. Perform the following steps:

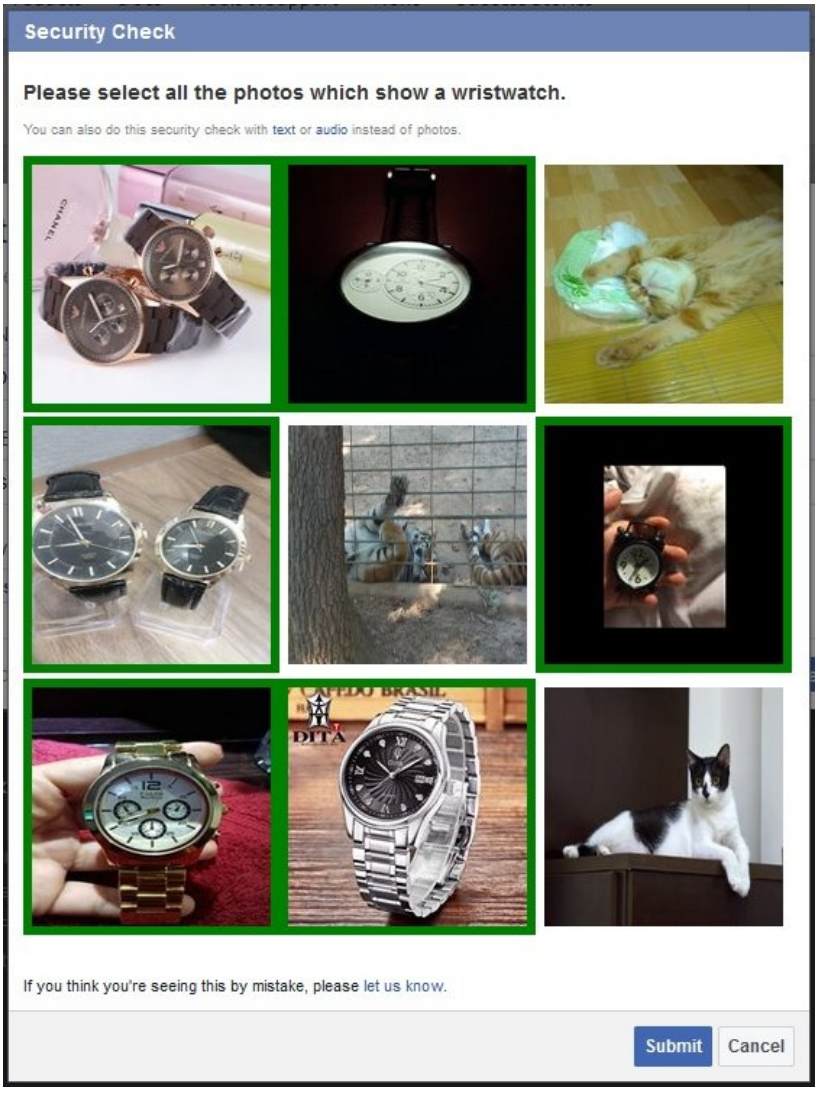
   a. Enter a name for the new Facebook application you are creating in the Display Name field.

b. Enter a valid email address in the Contact Email field.

c. Select a category for your Facebook application from the Category drop-down list.

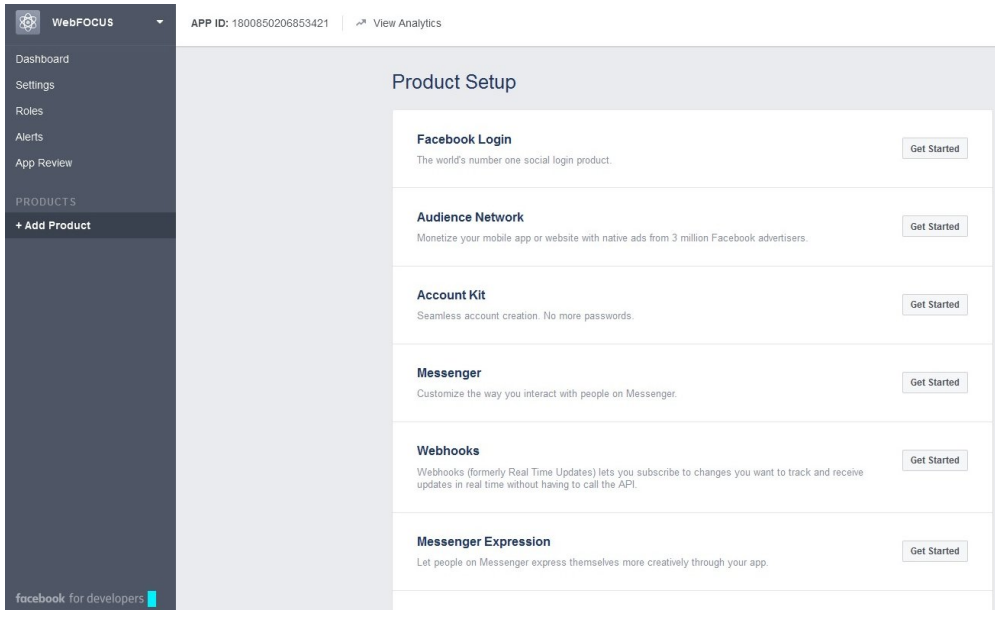
6. Click *Create App ID*.

   The Security Check screen is displayed, as shown in the following image.



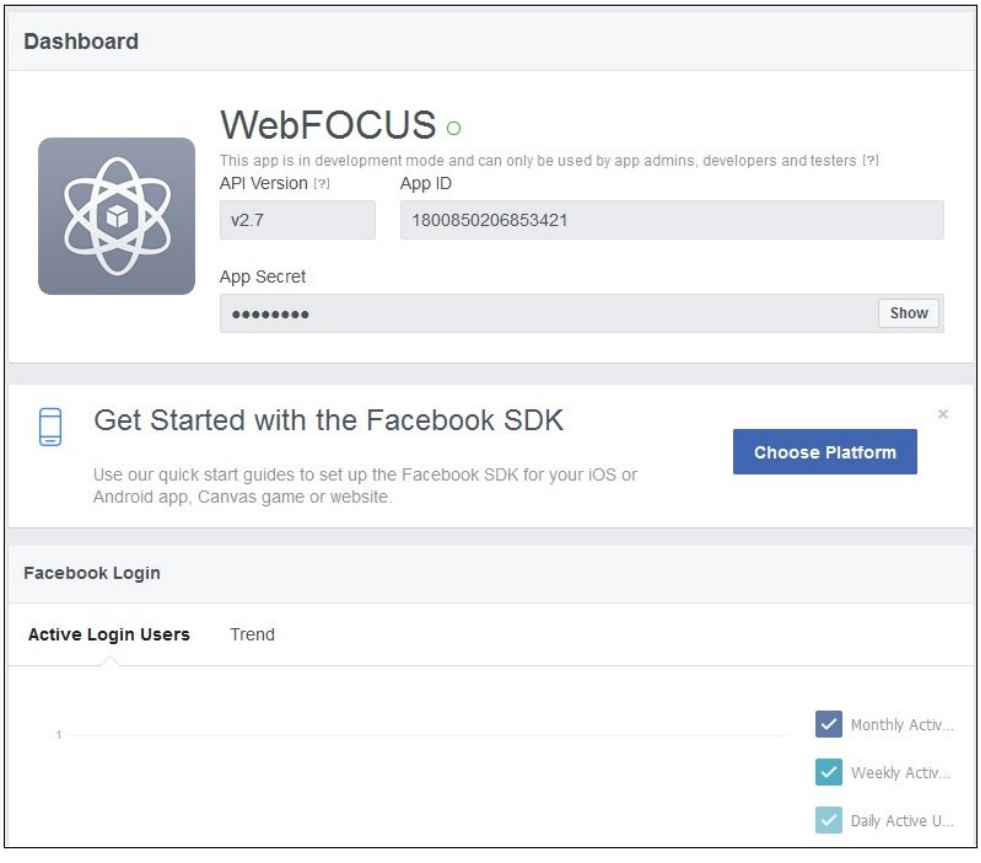
7. Click the required images as specified by the security instructions, and then click *Submit*.

The Product Setup screen is displayed, as shown in the following image.



8. Click *Dashboard* in the left pane.

The Dashboard screen is displayed and is branded based on the display name you specified in Step 5, as shown in the following image.



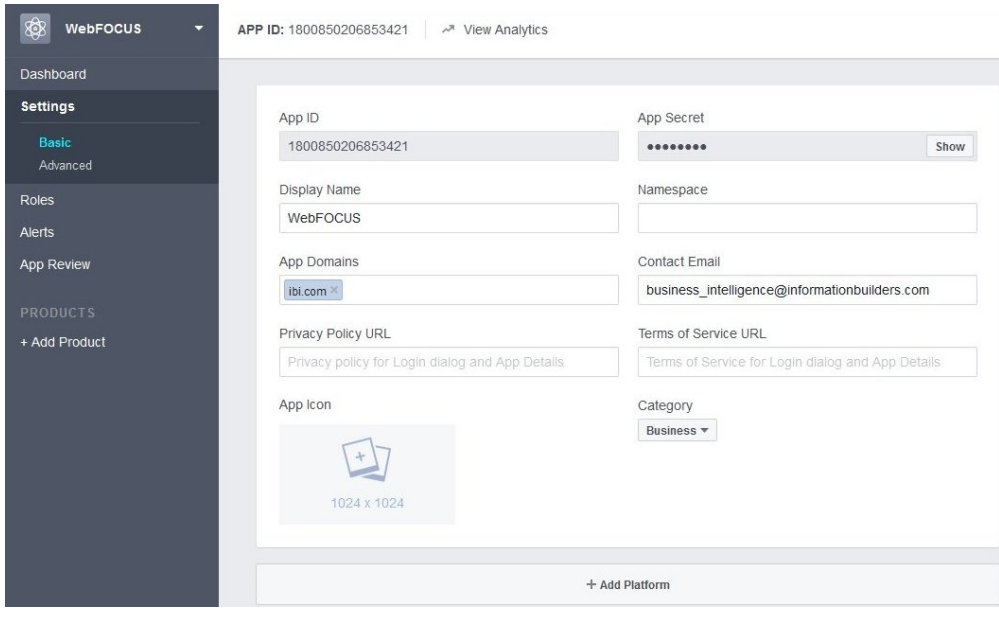
The Dashboard screen contains the App ID and App Secret values. The App Secret value is hidden by default. These values are required for configuring the Facebook Adapter.

9. To view the App Secret value, click *Show*.

   A prompt for the password of the Facebook user creating the application is displayed.

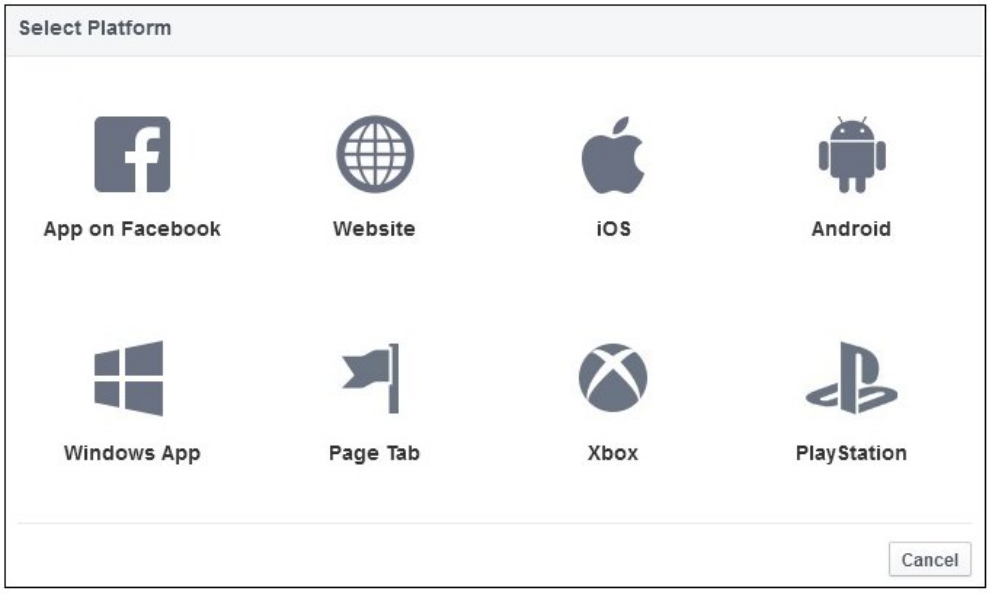
10. Enter the valid password and then click *Submit*.
11. Click *Settings* in the left pane.

The Settings screen is displayed, as shown in the following image.



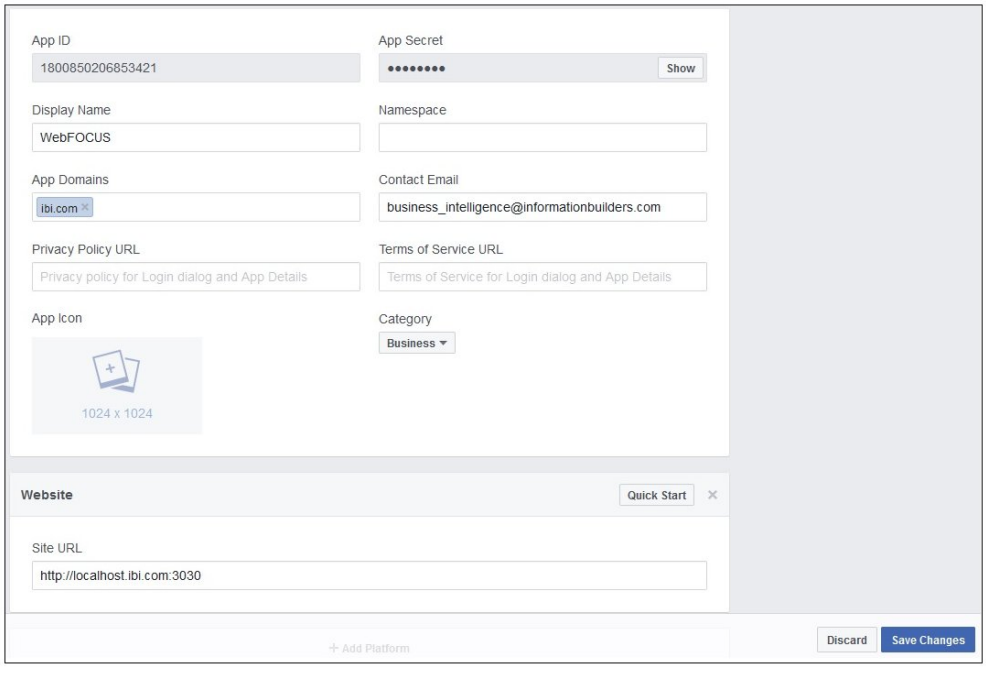
12. In the App Domains field, enter the domain where the WebFOCUS Reporting Server is installed.
13. Click + *Add Platform*.

The Select Platform screen is displayed, as shown in the following image.



14. Click *Website*.

You are returned to the Settings screen, as shown in the following image.



15. Enter a Site URL with the domain matching where the WebFOCUS Reporting Server is installed.
16. Click *Save Changes*.

    You are now ready to configure the Facebook Adapter.

Configuring the Facebook Adapter

This section describes how to configure the Facebook Adapter.

Procedure: How to Configure the Facebook Adapter

1. Clear the cookies from the web browser that will be used to start the WebFOCUS Reporting Server Web Console.
2. Access the WebFOCUS Reporting Server Web Console with a URL containing the domain where the WebFOCUS Reporting Server is installed. For more information, see *Creating a Facebook Application* on page 721.

   This domain should match the one configured with the Facebook application. For example:

   ```
   http://IBI-Computer.ibi.com:8121
   ```

3. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

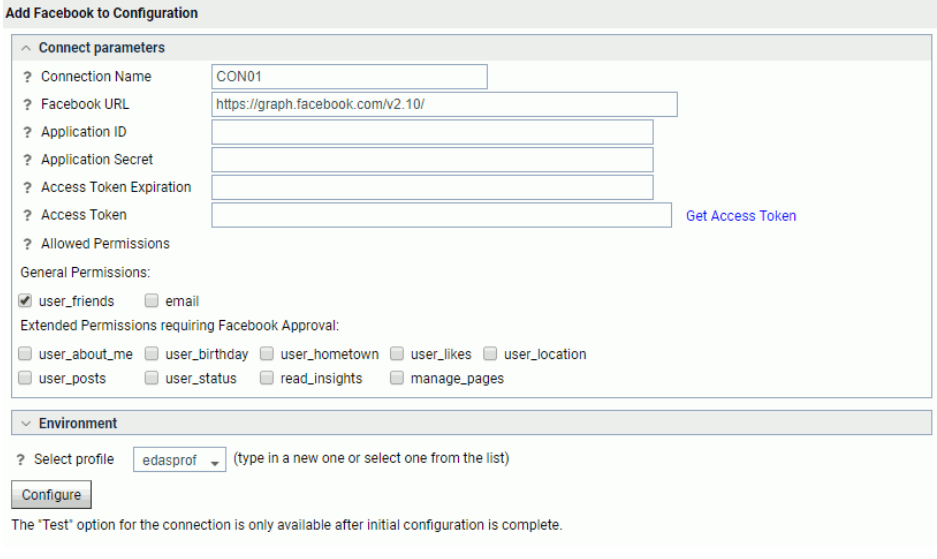
   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

4. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

5. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

6. Right-click the *Facebook* node and select *Configure*.

   The Add Facebook to Configuration pane opens, as shown in the following image.



7. Enter the values for the Application ID and Application Secret as defined in the Facebook application you created.

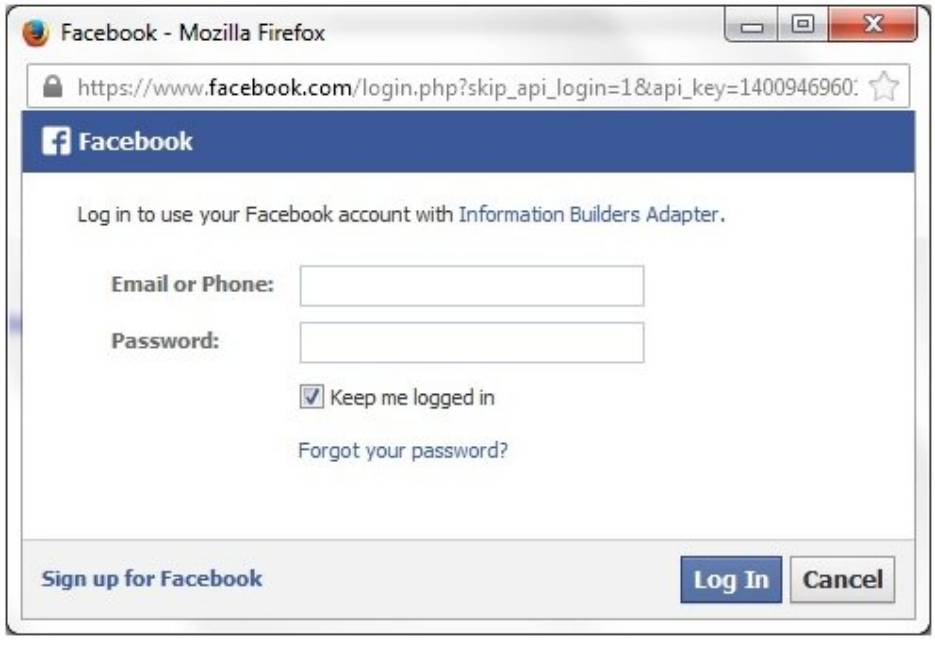
   For more information, see *Creating a Facebook Application* on page 721.

8. Choose the options in the General Permissions area and Extended Permissions that are to be granted to the Facebook Application. Note that the Extended Permissions require Facebook approval.

   Click the *Get Access Token* link.

For more information on the Allowed Permissions, see *Connection Attributes for Facebook* on page 733.

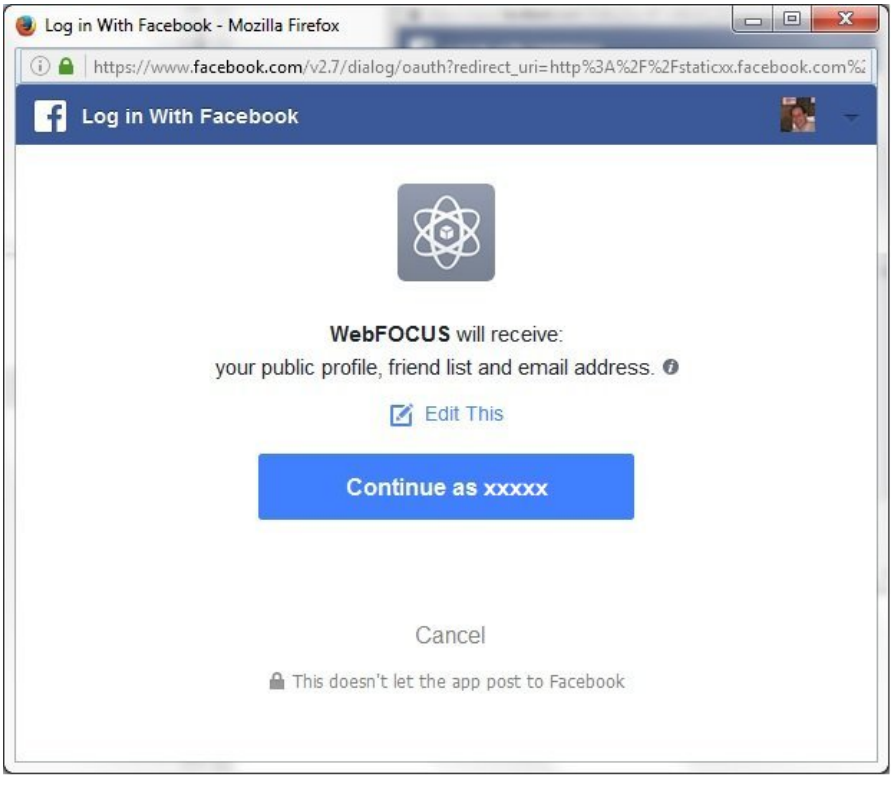
A Facebook login dialog opens, as shown in the following image.



If the Facebook login dialog does not open, ensure that the Pop-up Blocker within the web browser is either disabled or configured to allow pop-up windows from the host where the WebFOCUS Reporting Server is installed.

9. Enter the Facebook login credentials and then click *Log In*.

A pop-up window is displayed asking you to confirm the permissions that are to be granted to the Facebook application, as shown in the following image.



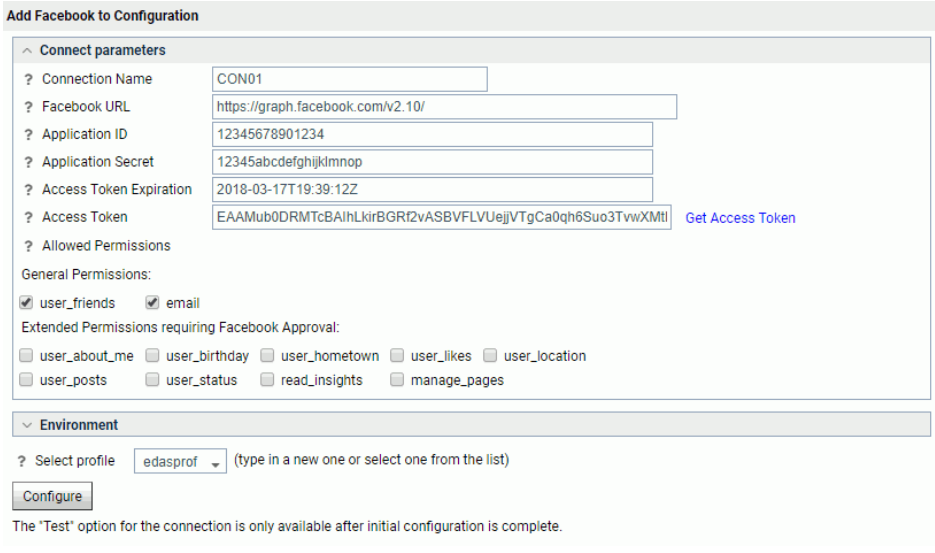
10. Click *Continue as xxxxx*.

    where:

    xxxxx

    Is the name of the Facebook user.

You are returned to the Add Facebook to Configuration pane where the Access Token Expiration and the Access Token fields are now populated, as shown in the following image.



11. Click *Configure*.

    The configured Facebook Adapter is added to the Facebook node in the left pane.

    Note: The Access Token expires after 60 days. To refresh the Access Token, click the *Get Access Token* link and then click *Configure*.

Reference: Connection Attributes for Facebook

The following list describes the connection attributes for the Facebook Adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

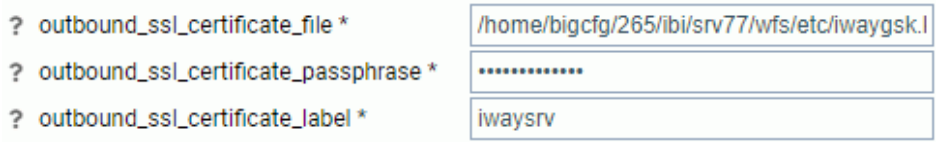
Facebook URL

The URL of the Facebook Graph API request. The default value is:

`https://graph.facebook.com/`

For iSeries machines, the WebFOCUS Reporting Server must be configured for SSL as follows:

1. From the Web Console sidebar, click *Workspace*.
2. From the menu bar, click *Workspace Set*, and select *Miscellaneous Settings*
3. Enter values for *outbound_ssl_certificate_file*, *outbound_ssl_certificate_passphrase*, and *outbound_ssl_certificate_label*, and then click *Save*. For example:



Application ID

The Facebook application ID as defined during the creation of the Facebook Application. For more information, see *Creating a Facebook Application* on page 721.

Application Secret

The Facebook application secret as defined in the Facebook application. For more information, see *Creating a Facebook Application* on page 721.

Access Token Expiration

The date and time that the Access Token will expire.

Access Token

Click the *Get Access Token* link to obtain this token. The credentials for a Facebook account are then entered. The value for the Access Token is returned by an authorized login.

General Permissions

Grants the selected permissions to the Facebook application, which include:

- ❑ user_friends
- ❑ email

Extended Permissions requiring Facebook Approval

Grants the selected permissions to the Facebook application, which include:

- ❑ user_about_me
- ❑ user_birthday

- user_hometown
- user_likes
- user_location
- user_posts
- user_status
- read_insights
- manage_pages

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the connection attributes. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), select *New Profile* from the drop-down list and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

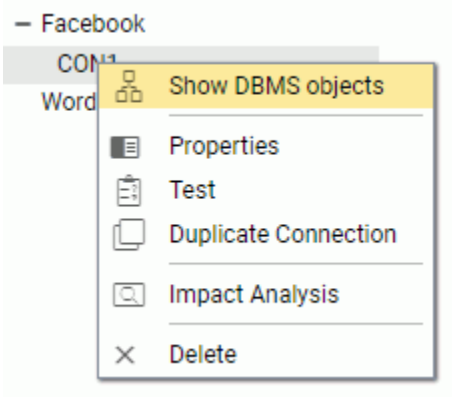
Creating Metadata and Sample Reports for the Facebook Adapter

Create Synonym for the Facebook Adapter creates the metadata used for WebFOCUS reporting and DataMigrator ETL flows. It also creates sample WebFOCUS reports and DataMigrator ETL flows.

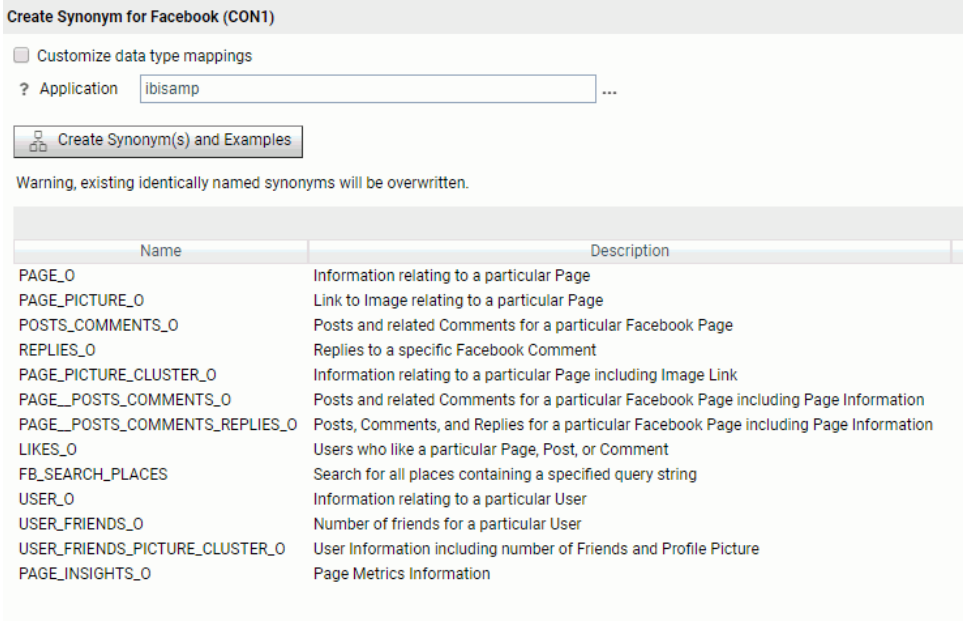
Procedure: How to Create Metadata and Sample Reports

1. From the WebFOCUS Reporting Server Web Console, expand the *Adapters* folder, *Configured* folder, and then the *Facebook* folder.

2. Right-click the configured connection for the Facebook Adapter (for example, FB) and select *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for Facebook pane opens, as shown in the following image.



Name	Description
PAGE_O	Information relating to a particular Page
PAGE_PICTURE_O	Link to Image relating to a particular Page
POSTS_COMMENTS_O	Posts and related Comments for a particular Facebook Page
REPLIES_O	Replies to a specific Facebook Comment
PAGE_PICTURE_CLUSTER_O	Information relating to a particular Page including Image Link
PAGE__POSTS_COMMENTS_O	Posts and related Comments for a particular Facebook Page including Page Information
PAGE__POSTS_COMMENTS_REPLIES_O	Posts, Comments, and Replies for a particular Facebook Page including Page Information
LIKES_O	Users who like a particular Page, Post, or Comment
FB_SEARCH_PLACES	Search for all places containing a specified query string
USER_O	Information relating to a particular User
USER_FRIENDS_O	Number of friends for a particular User
USER_FRIENDS_PICTURE_CLUSTER_O	User Information including number of Friends and Profile Picture
PAGE_INSIGHTS_O	Page Metrics Information

3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata, sample reports, and DataMigrator ETL flows are to be stored.
4. Click *Create Synonym(s) and Examples*.

The Create Synonym for Facebook Status pane opens and indicates that the synonym was created successfully.

Facebook Adapter Examples

This section describes the metadata and sample reports for the Facebook Adapter.

Reference: Facebook Adapter Metadata

The following table lists and describes the available metadata for the Facebook Adapter.

A metadata name with a "_o" suffix is used to retrieve an object of information from Facebook. This type of metadata should only be used as a secondary file in a JOIN.

A metadata name without a "_o" suffix is used to retrieve specific columns of information from Facebook.

Metadata	Description
fb_search	Used to search for a Facebook ID for a page or user.
fb_search_places	Used to search for places containing a specified query string.
likes_o	Used to retrieve the IDs and names of users who like a given video, post, comment, link, photo, or album.
page_insights_o	Used to retrieve specific statistics about a particular Facebook page.
page_o	Used to retrieve information about a Facebook page.
page_picture_o	Used to retrieve a picture for a Facebook page or user.
posts_comments_o	Used to retrieve Posts and Comments for a Facebook page.

Metadata	Description
profile_o	Used to retrieve the name, user name/page alias, and type for a Facebook object. For example, user, group, page, event, application.
replies_o	Used to retrieve replies for comments.
user_friends_o	Used to retrieve friends for a Facebook user.
user_o	Used to retrieve information about a user.
fbsampl/fb_comment_sentiment	Describes a SQL Server table loaded by the fb_datamigrator_sentiment_load DataMigrator flow. The table contains sentiment scores for Facebook comments. Used when the WAND Sentiment Analysis Adapter is configured.
fbsampl/fb_comments	Describes a SQL Server table loaded by the fb_datamigrator_load DataMigrator flow. The table contains Facebook comments.
fbsampl/fb_comments_to_sentiment	Cluster Join from fb_comments to wandscore; wandscore metadata is created from the WAND Sentiment Analysis Adapter. Used by the fb_datamigrator_sentiment_load DataMigrator flow.
fbsampl/ fb_join_datamodel_excluding_sentiment	Cluster Join used for reporting post, posters, comments, commenters, replies, and repliers. Joins: ❑ fb_posts to fb_profile ❑ fb_posts to fb_page_info ❑ fb_posts to fb_comments ❑ fb_comments to fb_profile ❑ fb_comments to fb_replies ❑ fb_replies to fb_profile

Metadata	Description
fbsampl/ fb_join_datamodel_including_sentiment	Cluster Join used for reporting post, posters, comments, commenters, replies, repliers, post sentiment score, comment sentiment score, and reply sentiment score. Joins: ❏ fb_posts to fb_profile ❏ fb_posts to fb_page_info ❏ fb_posts to fb_comments ❏ fb_comments to fb_profile ❏ fb_comments to fb_replies ❏ fb_replies to fb_profile ❏ fb_posts to fb_post_sentiment ❏ fb_comments to fb_comment_sentiment ❏ fb_replies to fb_comment_sentiment Used when the WAND Sentiment Analysis Adapter is configured.
fbsampl/fb_page_info	Describes a SQL Server table loaded by the fb_datamigrator_load DataMigrator flow. The table contains Facebook page information.
fbsampl/fb_post_sentiment	Describes a SQL Server table loaded by the fb_datamigrator_sentiment_load DataMigrator flow. The table contains sentiment scores for Facebook posts. Used when the WAND Sentiment Analysis Adapter is configured.
fbsampl/fb_posts	Describes a SQL Server table loaded by the fb_datamigrator_load DataMigrator flow. The table contains Facebook posts.

Metadata	Description
fbsampl/fb_posts_to_sentiment	Cluster Join from fb_posts to wandscore; wandscore metadata is created from the WAND Sentiment Analysis Adapter. Used by the fb_datamigrator_sentiment_load DataMigrator flow.
fbsampl/fb_profile	Describes a SQL Server table loaded by the fb_datamigrator_load DataMigrator flow. The table contains name, user name/page alias, and type information for a Facebook object. For example, user, group, page, event, application.
fbsampl/fb_replies	Describes a SQL Server table loaded by the fb_datamigrator_load DataMigrator flow. The table contains Facebook replies.
fbsampl/fb_replies_to_sentiment	Cluster Join from fb_replies to wandscore; wandscore metadata is created from the WAND Sentiment Analysis Adapter. Used by the fb_datamigrator_sentiment_load DataMigrator flow.
fbsampl/fb_user	Describes a SQL Server table loaded by the fb_datamigrator_load DataMigrator flow. The table contains user information.
fbsampl/page_insights_cluster	Cluster Join used for reporting page and page statistics. Joins: page_o to page_insights_o
page_picture_cluster_o	Cluster Join used for reporting page and page picture statistics. Joins: page_o to page_picture_o

Metadata	Description
fbsampl/page_posts_comments_o	Cluster Join used for reporting page posts and comments. Joins: page_o to posts_comments_o
fbsampl/posts_comments_replies_o	Cluster Join used for reporting page, posts, comments, and replies. Joins: ❏ page_o to posts_comments_o ❏ posts_comments_o to replies_o
fbsampl/ user_friends_picture_cluster_o	Cluster Join for reporting user information, number of friends, and user picture. Joins: ❏ user_o to user_friends_o ❏ user_o to page_picture_o
fbsampl/ user_picture_cluster_o	Cluster Join for reporting user information and user picture. Joins: user_o to page_picture_o

Reference: Facebook Adapter Sample Reports

The following table lists and describes the sample reports for the Facebook Adapter.

Sample Report	Description
fbsampl/fb_create_datamodel	Creates the SQL Server tables used for the Facebook data model loaded by the fb_datamigrator_load and fb_datamigrator_sentiment_load DataMigrator flows. As a prerequisite, a SQL Server connection called *socialmedia* must be configured as well as the creation of a SQL Server database called *Facebook*.
fbsampl/fb_delete_datamodel	Deletes the SQL Server tables used for the data model.
fbsampl/fb_page_drill	Page information drill report. Uses: fbsampl/fb_page_info
fbsampl/fb_page_posts_comments_replies_report	Reports Facebook page posts with related comments and replies. Uses: fbsampl/fb_join_datamodel_excluding_sentiment
fbsampl/fb_page_posts_comments_replies_scored_report	Reports Facebook page posts with related comments and replies including sentiment scoring. Uses: fbsampl/fb_join_datamodel_including_sentiment
fbsampl/fb_search	Search for the ID and name for a page or user. Uses fb_search.

Sample Report	Description
fbsampl/fb_search_places	Search for places containing a specified query string. Uses: fb_search_places
fbsampl/fb_tagcloud_drill_excluding_sentiment	Drill to posts, comments, and replies from the tag cloud. The Words Analysis adapter must be configured. Uses: fbsampl/fb_join_datamodel_excluding_sentiment
fbsampl/fb_tagcloud_drill_including_sentiment	Drill to posts, comments, replies, and sentiment from the tag cloud. The Words Analysis adapter must be configured. Uses: fbsampl/fb_join_datamodel_including_sentiment
fbsampl/fb_tagcloud_drill_post_excluding_sentiment	Reports on a specific post with related comments and replies. The Words Analysis adapter must be configured. Uses: fbsampl/fb_join_datamodel_excluding_sentiment
fbsampl/fb_tagcloud_drill_post_including_sentiment	Reports on a specific post with related comments, replies, and sentiment. The Words Analysis adapter must be configured. Uses: fbsampl/fb_join_datamodel_including_sentiment

Sample Report	Description
fbsampl/fb_tagcloud_posts_comments_replies	Tag cloud graph for posts, comments, and replies. The Words Analysis adapter must be configured. Uses: fbsampl/fb_join_datamodel_excluding_sentiment wan_document (Words Analysis metadata)
fbsampl/fb_user_drill	User information drill report. Uses: fbsampl/fb_user
fbsampl/page_insights_o	Reports metrics for a specific page. Uses: fbsampl/page_insights_cluster_o
page_o	Reports page information for a specific page. Uses: fbsampl/page_picture_cluster_o
fbsampl/page_posts_comments_o	Reports Facebook page posts and related comments. Uses: fbsampl/page_posts_comments_o
fbsampl/page_posts_comments_replies_o	Reports Facebook page posts and related comments including replies. Uses: fbsampl/page_posts_comments_replies_o
fbsampl/page_posts_likes_o	Reports users who liked particular posts. Uses: fbsampl/page_posts_comments_o and likes_o

Sample Report	Description
fbsampl/profile_o	Reports the type for an Object ID (Page or User). Uses: profile_o
fbsampl/user_o	Reports user information for a Facebook user. Uses: fbsampl/user_friends_picture_cluster_o

Reference: **Facebook Adapter DataMigrator Flows**

The following table lists and describes the DataMigrator flows for the Facebook Adapter. The Facebook Data Model must first be created by running fbsampl/fb_create_datamodel.

Flow	Description
fbsampl/fb_datamigrator_load	Process flow to run the following data flows: ❑ fbsampl/fb_load_page ❑ fbsampl/fb_load_posts ❑ fbsampl/fb_load_comments_replies ❑ fbsampl/fb_load_profile_posters ❑ fbsampl/fb_load_profile_commenters ❑ fbsampl/fb_load_profile_repliers ❑ fbsampl/fb_load_pages ❑ fbsampl/fb_load_users

Flow	Description
fbsampl/ fb_datamigrator_sentiment_load	Process flow to run the following data flows: ❑ fbsampl/fb_load_post_sentiment ❑ fbsampl/fb_load_comment_sentiment ❑ fbsampl/fb_load_reply_sentiment
fbsampl/ fb_load_comment_sentiment	Data flow to load sentiment scores for Facebook comments.
fbsampl/fb_load_comments_replies	Data flow to load Facebook comments and replies.
fbsampl/fb_load_page	Data flow to load Facebook page information for pages identified in the fbsampl/page_ids.ftm file.
fbsampl/fb_load_pages	Data flow to load Facebook page information for people that have either posted, commented, or replied from a Facebook page.
fbsampl/fb_load_post_sentiment	Data flow to load sentiment scores for Facebook posts.
fbsampl/fb_load_posts	Data flow to load Facebook posts.
fbsampl/ fb_load_profile_commenters	Data flow to load Facebook user and page profile information for commenters. For example, ID, name, and commenter type (page/user).
fbsampl/fb_load_profile_posters	Data flow to load Facebook user and page profile information for posters. For example, ID, name, and poster type (page/user).
fbsampl/fb_load_profile_repliers	Data flow to load Facebook user and page profile information for repliers. For example, ID, name, and replier type (page/user).
fbsampl/fb_load_reply_sentiment	Data flow to load sentiment scores for Facebook replies.

Flow	Description
fbsampl/fb_load_users	Data flow to load Facebook user information for people that have either posted, commented, or replied.

Chapter 27 Using the Adapters for Fixed-Format and Delimited Files

The Adapters for Fixed-Format and Delimited Flat Files access sequential files, where records are stored and retrieved in the same order as they are entered. Sequential data sources may be:

- Fixed-format files (identified as flat files in prior releases), in which each field occupies a predefined position in the record. (To maintain the fixed structure, unoccupied positions are filled by blanks.)
- Delimited-format files (identified as delimited files in this chapter), in which fields are stored in a predefined sequence, separated (that is, delimited) by a special character. (Although fields must maintain their predefined order in the file, blanks are not required to represent unoccupied positions.)

The adapters convert application requests into input/output (I/O) calls and return optimized answer sets to the requesting application.

You can also access compressed data sets with this adapter. However, the actual compression is done using the ZCOMP Exit. For details, see *Data Set Compression Exit: ZCOMP* on page 2719.

In this chapter:

Preparing the Fixed-Format and Delimited File Environment

The Adapters for Fixed-Format and Delimited Files do not require setting any environment variables.

Configuring the Adapters for Fixed-Format and Delimited Files

You can configure the Adapters for Fixed-Format and Delimited Files from the Web Console or the Data Management Console. In addition, for each type of file, you must specify the location of the data files you wish to access. You can define the location, either by:

- Including a FILEDEF command in a supported server profile. This approach enables you to quickly change the physical file name associated with the metadata.

- Specifying the location when you create a synonym. This approach enables you to define the data file location in the Master File.

Procedure: How to Configure the Adapters for Fixed-Format and Delimited Files

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Note: The Fixed-Format Files and Delimited Files adapters are under the *Sequential and Indexed* group folder. No input parameters are required.

Procedure: How to Configure a Connection to an FTP Server for Delimited and Fixed-Format Files From the Web Console or the Data Management Console

An FTP Server can be used as a source for delimited and flat files by configuring a connection to the server. First, you must configure an adapter.

To configure a connection to an FTP server:

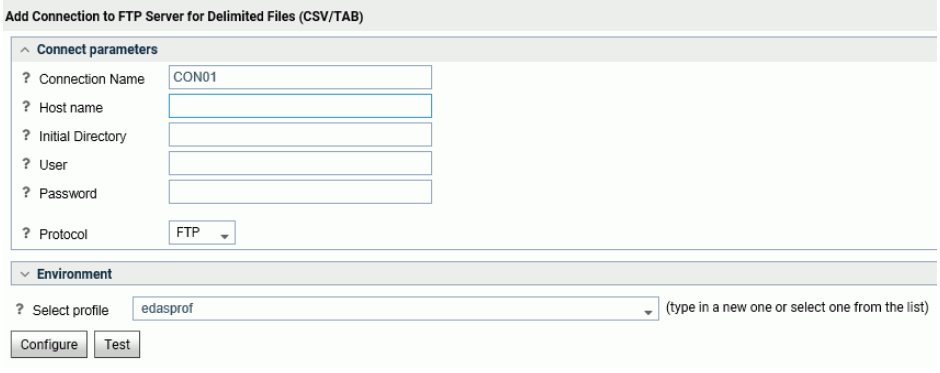
1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. Right-click the *Fixed-Format Files* or *Delimited Files* folder and select *Add Connection*.
3. The Add Connection to FTP Server for... window opens



4. Enter the Connection Parameters. See the *FTP Connection Parameters* below for details.
5. Click *Configure*.

Reference: FTP Connection Parameters

These are the connection parameters for FTP.

Parameter	Value
Connection Name	Name for this connection, default is CON01 **Note:** To use the Connection Name, you need to specify it in the CONNECTION attribute of the synonym located in the Data Allocation section.
Host Name	Name or IP address of FTP Server
Initial Directory	Directory on FTP server on initial connection
User	User ID to connect to FTP server or anonymous
Password	Password to connect to FTP or an email address

Parameter	Value
Profile	Profile on server to add this connection

Syntax: How to Specify the Location of a Fixed-Format or Delimited File on UNIX, Windows, or in an IBM i IFS or z/OS HFS File System

FIXED data sets may be allocated with the FILEDEF command

```
FILEDEF FILE1 DISK filename [(LRECL lrecl RECFM recfm]
```

where:

filename

Is the full path and file name of the data file.

lrecl

Is the record length in bytes. This parameter is optional. If you omit it, it is calculated based on the metadata description.

The left parenthesis preceding the optional parameters is required.

recfm

Is the record format. Specify F for fixed format, V for variable format. This parameter is optional. If you omit it, the default is fixed format.

Syntax: How to Specify the Location of a Fixed-Format or Delimited File in a z/OS PDS or Data Set

FIXED data sets may be allocated using JCL

```
//FILE1 DD DISP=SHR, DSN=dataset_name
```

or file definition

```
FILEDEF FILE1 DISK //'dataset_name' [(LRECL lrecl RECFM recfm]
```

where:

dataset_name

Is the fully qualified data set name.

Managing Metadata for Fixed-Format and Delimited Files

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Fixed-Format or Delimited Files data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Fixed-Format or Delimited File that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server metadata.

Creating Synonyms for Fixed-Format Files

A Fixed-Format file is a sequential file in which each field occupies a predefined position in the record, with unoccupied positions filled by blanks to maintain the fixed structure.

Procedure: How to Create a Synonym for a Fixed-Format File

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Web Console or the Data Management Console.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the synonym creation parameters reference.
   - For Windows, UNIX, and IBM i, see *Synonym Creation Parameters for Fixed-Format Files on Windows, UNIX, and IBM i IFS File Systems* on page 754.
   - For z/OS, see *Synonym Creation Parameters for Fixed-Format Files on z/OS* on page 758.
4. After entering the parameter values, click *Create Synonym*.

   The Status pane indicates that the synonym was created successfully.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the *Validate* check box, the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Fixed-Format Files on Windows, UNIX, and IBM i IFS File Systems

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Cobol FD

Path to the directory containing multiple COBOL files.

Customize COBOL FD conversion options

Select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select this check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Select Data Files Allocation

Choose one of two options:

- **Single File** defines a single data file. This option is not applicable for remote connections.
- **Collection of files** defines a collection of data files to be described by one synonym.

Select Data Processing Mode

Indicates the data retrieval type.

- **One-time retrieval** retrieves data once in order to create synonym for sequential file.
- **File Listener** provides continued retrieval.

File Listener attributes

Listening directory polling interval for new files

Indicates how often the listener will check for the presence of a file in number of seconds. The default is 10 seconds.

Timeout interval in listening directory for new files

Indicates the timeout interval for the listener in number of seconds. The default is 10 seconds.

Maximum number of records processed in the request

Indicates the number of records processed within one request. This option is available for single file processing.

Maximum number of data files processed in the request

Indicates number of files processed within one request. This option is available for processing collection of files. The default is 99,999,999.

Data Files Pickup Strategy

Indicates the mechanism used by files agents to pick up files from their directories.

- **IMMEDIATE** processes the file as soon as it is detected.
- **TRIGGER** processes the file as soon as a trigger file is detected. This option requires an additional setting for the trigger file extension.

  Note: In OpenVMS, a dot (.) cannot be a part of a file name (that is, the file name prior to the extension). If the trigger file name for a listener includes a dot, the dot will be replaced with an underscore (_). For example, if a user enters file1.ext1.trig for the trigger file name, DataMigrator will replace it with file1_ext1.trig.

Trigger File Extension

Indicates the extension of the trigger file. The trigger file extension is added to the full name of the file being listened for.

Processed Data Files Discard Strategy

Indicates how the file is handled after it has been processed by a file agent.

- **DELETE** deletes the file.
- **ARCHIVE** copies the file with the timestamp to a specified directory defined in *File Listener Archiving Directory* option.
- **KEEP** keeps the files.

File Listener Archiving Directory

Defines the location where the data files will be archived when DISCARD strategy of ARCHIVE is used. This can be an application directory, a mapped application name, or a directory location on your file system.

When using a directory on a file system for local files, the name of the physical directory can be used.

When using a directory on a file system for remote data files, a directory relative to the initial directory can be used. For example, if your remote initial directory is /home/user1/apps, using *sales* will set the data files directory location to /home/user1/apps/sales.

Collection Definition attributes

Directory or Application

Defines the location of data files for polling. This can be an application directory, a mapped application name, or a directory location on your file system.

When using a directory on a file system for local files, the name of the physical directory can be used.

When using a directory on a file system for remote data files, a directory relative to the initial directory can be used. For example, if your remote initial directory is /home/user1/apps, using *sales* will set the data files directory location to /home/user1/apps/sales.

File Selection Mask attributes

These attributes allow you to limit the list of data files.

Name

Type a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.

Extension

Type an extension with or without the wildcard symbol %.

Synonym field names processing options

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Synonym Name

Indicates the name of the synonym.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Extended data format attributes

Codepage

This option provides data portability between servers by enabling processing of data loaded in different encoding systems.

Click the *Extended data format attributes* check box to display the *Codepage* option.

In the input box provided, enter the code page of the stored data. Your entry is added to the Master File of the generated synonym.

Note: The code page must have a customized conversion table as part of your NLS configuration. If you have not already requested a conversion table for the designated code page, you can do so now. On the Web Console menu bar, click *Workspace*, then *Configuration*, and choose *NLS* under General settings. The NLS Configuration Wizard opens. Click *Customize code page conversion tables* and select the check box for the code page you wish to use. Click the *Save and Restart* button to implement your new setting.

Reference: Synonym Creation Parameters for Fixed-Format Files on z/OS

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

File System Selection

Choose one of the following options from the drop-down menu:

- *Fully qualified PDS name* to indicate a partitioned data set on MVS.

  In the input boxes provided, type a PDS name preceded by // and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

- *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

  In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and a File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).

Synonym field name processing options

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize options

Select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Extended data format attributes

Codepage

This option provides data portability between servers by enabling processing of data loaded in different encoding systems.

Click the *Extended data format attributes* check box to display the *Codepage* option.

In the input box provided, enter the code page of the stored data. Your entry is added to the Master File of the generated synonym.

Note: The code page you specify here must be one for which you have requested a customized code page conversion table as part of your NLS configuration. If you have not already requested a conversion table for the designated code page, you can do so now. On the Web Console menu bar, click *Workspace*, then *Configuration*, and choose *NLS* under General settings. The NLS Configuration Wizard opens. Click *Customize code page conversion tables* and select the check box for the code page you wish to use. Click the *Save and Restart* button to implement your new setting.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select files

Select files for which you wish to create synonyms:

- To select all files in the list, click the *Select All* button.
- To select specific files, select the corresponding check boxes.

Choose Data File Name

After choosing files, choose the corresponding data file name from the drop-down list.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Creating Synonyms for Delimited Files

A delimited file is a sequential file in which fields are stored in a predefined sequence, separated (that is, delimited) by a special character or characters. (Although fields must maintain their predefined order in the file, and delimiters are required, blanks are not required to represent unoccupied positions.

The types of field delimiters are:

- Comma.
- Tab.
- | (pipe)
- Token, which is any combination of characters defined in the data source Synonym. (Note that token-delimited files cannot be used in Joins.)

You can create different types of synonyms using the Adapter for Delimited Files:

- A synonym for a file that resides on a server or on a local machine.
- A synonym for a collection of files with a one-time retrieval.
- A synonym for a collection of files for a File Listener.
- Multiple Synonyms.

Procedure: How to Create a Synonym for a Delimited File

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Web Console or the Data Management Console.

1. From the Web Console menu bar, click *Applications*, then *New,* and then *Synonym*

   or

   From the Data Management Console, right-click an application directory, then select *New*, and then *Synonym*.

   The *Select Data File for Delimited Flat File Synonym* pane opens.
2. Enter values for the parameters required by the adapter as described in the synonym creation parameters reference.
3. After entering the parameter values, click *Create Synonym*.

   The Status pane indicates that the synonym was created successfully.

The synonym is created and added under the specified application directory.

Note: You can also create a synonym from the Adapters page by right-clicking a configured adapter connection and selecting *Create Synonym*.

Reference: Synonym Creation Parameters for Delimited Files

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the *Next* button, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Create Multiple Synonyms

Select this check box for Multiple Synonyms.

Data Files location

Is the path to multiple data files.

File location

Defines the location of the data file.

From the drop-down menu select:

- **Server side.**
- **Local machine.**

Data File

Defines an absolute path to the data file on the server or a directory relative to the initial one. It includes the name of the data file.

Enter the path and file name or click the ellipsis (...) to select the file.

Upload file

Click the ellipsis (...) to select the data file to be uploaded.

Select Data Files Allocation

Select:

- **Single File** to define a single data file. This option is not applicable for remote connections.
- **Collection of files** to define a collection of data files to be described by one synonym.

Select Data Processing Mode

Indicates the data retrieval type:

- **One-time retrieval** retrieves data once, in order to create a synonym for a sequential file.

- **File Listener** provides continual retrieval.

Record Specification

Record Delimiter

Defines a character or series of characters that indicate the separation between file records or data rows (for example, groups of delimited fields).

This attribute is reflected in the Access File as RDELIMITER.

Non-printable Record Delimiter

Defines whether the string in the delimiter field is a printable string or a non-printable decimal character. Please note that a non-printable decimal character will appear (translated) in the Access file as a hexadecimal value. If the delimiter is entered as hexadecimal, set this attribute to *No*.

Field Specification

Delimiter

Defines a character or series of characters that indicate the separation between file fields.

To use multiple characters, enter them together. For example enter two slashes (//).

To use a tab character as a delimiter, enter the reserved word TAB.

Alternatively, you can enter hexadecimal values directly. For example, Linefeed and Carriage Return would be 0x0A and 0x0D, respectively.

If several characters form a separator, enter then as one string. For example, Linefeed and Carriage Return would be 0x0A0D.

Non-printable Delimiter

Defines whether the string in the delimiter field is a printable string or a non-printable decimal character. Please note that the non-printable decimal character will appear (translated) in the Access file as a hexadecimal value.

If the delimiter is entered as a hexadecimal value, set this attribute to *No*.

If this attribute is set to *Yes*, then the values in the *Delimiter* attribute should be entered as decimals. For example, for non-printable separators such as Linefeed, use the decimal value of 10. If several non-printable characters form a separator, use a space between the decimal values. For example, to use Linefeed and Carriage Return together, enter 10 or 13.

Header Row

Select this check box to request that names in the header line be used as column names in the synonym.

`Number of first rows to skip`

Specifies the number of rows above the header row that should be ignored when creating the synonym and reading the data.

Enclosure

If character columns are enclosed in quotation marks or another symbol, enter an enclosure symbol for the character column format used in the delimited file(s) for which the synonyms are being generated. For example, for a file in which character columns are enclosed in double quotation marks, specify " in the *Enclosure* field.

Record Scan Limit

Enter a value to indicate how many records from the DFIX data file to scan to determine the length of the columns.

If this parameter is omitted, all records from the file are processed to make this determination.

Preserve format

Indicates whether the original data layout, including empty records and linefeeds, will be preserved.

- *No*. The original characters will be replaced by a printable string. This is the default.
- *Yes*. The original characters, including non-printable characters and linefeeds, will be preserved.

This attribute is reflected in the Access File as PRESERVEFRMT.

File Listener

Listening directory polling interval for new files

Indicates how often the listener will check for the presence of a file in number of seconds. The default is 10 seconds.

Timeout interval in listening directory for new files

Indicates the timeout interval for the listener in number of seconds. The default is 10 seconds.

Maximum number of data files processed in the request

Indicates number of files processed within one request. The default is 99,999,999.

Data Files Pickup Strategy

Indicates the mechanism used by files agents to pick up files from their directories.

- **IMMEDIATE** processes the file as soon as it is detected.
- **TRIGGER** processes the file as soon as a trigger file is detected. This option requires an additional setting for the trigger file extension.

  Note: In OpenVMS, a dot (.) cannot be a part of a file name (that is, the file name prior to the extension). If the trigger file name for a listener includes a dot, the dot will be replaced with an underscore (_). For example, if a user enters file1.ext1.trig for the trigger file name, DataMigrator will replace it with file1_ext1.trig.

Trigger File Extension

Indicates the extension of the trigger file. The trigger file extension is added to the full name of the file being listened for.

Processed Data Files Discard Strategy

Indicates how the file is handled after it has been processed by a file agent.

- **DELETE** deletes the file.
- **ARCHIVE** copies the file with the timestamp to a specified directory defined in File Listener Archiving Directory option.
- **KEEP** keeps the files.

File Listener Archiving Directory

Archiving directory on the server where the data file will be archived. You can use an absolute path or a directory relative to the initial one.

Collection Definition

Directory or Application

Defines the location type of the data files. It can be a physical directory for local files or a directory relative to the initial directory for remote data files. For example, if your remote initial directory is /home/user1/apps, using sales in collection definition will set the data files location to /home/user1/apps/sales.

File Selection Mask

Name

Type a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.

Extension

Type an extension with or without the wildcard symbol %.

Synonym field names processing options

Application

Select an application directory. The default value is *baseapp*.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select files

Select the check boxes to left of *Default Synonym Name* to choose the files for which you want to create synonyms. The data files are highlighted when you choose the corresponding check boxes.

Click *Select All Files* to select all the data files in the directory path.

Extended data format attributes

Codepage

This option provides data portability between servers by enabling processing of data loaded in different encoding systems.

Select the *Extended data format attributes* check box to display the Codepage option.

In the input box provided, enter the code page of the stored data. Your entry is added to the Master File of the generated synonym.

Note: The code page you specify here must be one for which you have requested a customized code page conversion table as part of your NLS configuration. If you have not already requested a conversion table for the designated code page, you can do so now. From the Web Console *Workspace* menu, select *Configuration/Monitor*. In the navigation pane, right-click *NLS* and select *Configure*. The NLS Configuration Wizard opens. Click *Customize code page conversion tables* and select the check box for the code page you wish to use. Click *Save and Restart Server* to implement your new setting.

Example: Sample DFIX Data File and Generated Synonym

This example is for a single synonym for a file on the serve.

DFIX data file (Comma-delimited):

```
ENGLAND,JAGUAR,V12XKE AUTO,CONVERTIBLE,2,7427,8878.0
ENGLAND,JAGUAR,XJ12L AUTO,SEDAN,5,11194,13491,12000
ENGLAND,JENSEN,INTERCEPTOR III,SEDAN,4,14940,17850,0
ENGLAND,TRIUMPH,TR7,HARDTOP,2,4292,5100,0
JAPAN,DATSUN,B210 2 DOOR AUTO,SEDAN,4,2626,3139,4300
JAPAN,TOYOTA,COROLLA 4 DOOR DIX AUTO,SEDAN,4,2886,3339,35030
ITALY,ALFA ROMEO,2000 4 DOOR BERLINA,SEDAN,4,4915,5925,4800
ITALY,ALFA ROMEO,2000 GT VELOCE,COUPE,2,5660,6820,12400
ITALY,ALFA ROMEO,2000 SPI,2,25000,31500,0,13000
ITALY,MASERATI,DORA 2 DOOR,COUPE,2,25000,31500,0
GERMANY,AUDI,100 LS 2 DOOR AUTO,SEDAN,5,5063,5970,7800
GERMANY,BMW,2002 2 DOOR,SEDAN,5,5800,5940,8950
GERMANY,BMW,2002 2 DOOR AUTO,SEDAN,4,6000,6355,8900
GERMANY,BMW,3.0 SI 4 DOOR,SEDAN,5,10000,13752,14000
GERMANY,BMW,3.0 SI 4 DOOR AUTO,SEDAN,5,11000,14123,1894
GERMANY,BMW,530I 4 DOOR,SEDAN,5,8300,9097,14000
GERMANY,BMW,530I 4 DOOR AUTO,SEDAN,5,8400,9495,1560
FRANCE,PEUGEOT,504 4 DOOR,SEDAN,5,4631,5610,0
```

To generate a synonym for the DFIX comma delimited data file, enter the following information on the Create Synonym panes of the Web Console or the Data Management Console:

1. In the Directory path field, enter the directory name containing the delimited flat file.
2. Enter the file name (for example, dfix_car).
3. Enter the file extension field (for example, .dat.)
4. Click *Select Files*.
5. On the second Create Synonym pane, select a file from the list of displayed data files and specify *comma*, as the separator, in the Delimiter field.
6. Click *Create Synonym*. The synonym is created and added under the specified application directory (baseapp is the default).

   A status window displays the message: *All Synonyms Created Successfully*

7. From the message window, click *Applications* on the menu bar.
8. Open the *baseapp* application folder in the navigation pane and click the synonym–in this example, *dfix_car*.
9. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File:

```
FILENAME=CAR_DFIX, SUFFIX=DFIX    ,
 DATASET=directory/car_dfix.dat, $
  SEGMENT=CAR_DFIX, SEGTYPE=S0, $
    FIELDNAME=FIELD_1, USAGE=A7, ACTUAL=A7BV, $
    FIELDNAME=FIELD_2, USAGE=A10, ACTUAL=A10BV, $
    FIELDNAME=FIELD_3, USAGE=A23, ACTUAL=A23BV, $
    FIELDNAME=FIELD_4, USAGE=A11, ACTUAL=A11BV, $
    FIELDNAME=FIELD_5, USAGE=I1, ACTUAL=A1, $
    FIELDNAME=FIELD_6, USAGE=I5, ACTUAL=A5V, $
    FIELDNAME=FIELD_7, USAGE=I5, ACTUAL=A5V, $
    FIELDNAME=FIELD_8, USAGE=I5, ACTUAL=A5V, $
```

Note that the FILE declaration in the generated Master File includes the attribute:

```
SUFFIX=DFIX
```

Generated Access File:

```
SEGNAME=DFIX_CAR, DELIMITER=',', HEADER=NO,$
```

The delimiter consists of a single printable character.

Example: Sample DFIX Data File With Header and Enclosure

This sample file contains a header line and uses the enclosure character ".

```
"COUNTRY","CAR","MODEL","BODYTYPE","SEATS","DEALER_COST","RETAIL_COST","SALE
S"
"ENGLAND","JAGUAR","V12XKE AUTO","CONVERTIBLE",2,7427,8878,0
"ENGLAND","JAGUAR","XJ12L AUTO","SEDAN",5,11194,13491,12000
"ENGLAND","JENSEN","INTERCEPTOR III","SEDAN",4,14940,17850,0
"ENGLAND","TRIUMPH","TR7","HARDTOP",2,4292,5100,0
"FRANCE","PEUGEOT","504 4 DOOR","SEDAN",5,4631,5610,0
"ITALY","ALFA ROMEO","2000 4 DOOR BERLINA","SEDAN",4,4915,5925,4800
"ITALY","ALFA ROMEO","2000 GT VELOCE","COUPE",2,5660,6820,12400
"ITALY","ALFA ROMEO","2000 SPIDER VELOCE","ROADSTER",2,5660,6820,13000
"ITALY","MASERATI","DORA 2 DOOR","COUPE",2,25000,31500,0
"JAPAN","DATSUN","B210 2 DOOR AUTO","SEDAN",4,2626,3139,43000
"JAPAN","TOYOTA","COROLLA 4 DOOR DIX AUTO","SEDAN",4,2886,3339,3503
"GERMANY","AUDI","100 LS 2 DOOR AUTO","SEDAN",5,5063,5970,7800
"GERMANY","BMW","2002 2 DOOR","SEDAN",5,5800,5940,8950
"GERMANY","BMW","2002 2 DOOR AUTO","SEDAN",4,6000,6355,8900
"GERMANY","BMW","3.0 SI 4 DOOR","SEDAN",5,10000,13752,14000
"GERMANY","BMW","3.0 SI 4 DOOR AUTO","SEDAN",5,11000,14123,18940
"GERMANY","BMW","530I 4 DOOR","SEDAN",5,8300,9097,14000
"GERMANY","BMW","530I 4 DOOR AUTO","SEDAN",5,8400,9495,15600
```

On the Synonym Creation panes, specify the *Delimiter* character as comma (,), the *Enclosure* character as double quotation marks ("), and check the *Header* box to use the names in the header line as field names (columns) in the Master File.

Generated Master File:

```
FILENAME=CAR_CSV, SUFFIX=DFIX    ,
 DATASET=directory/car_csv.csv, $
  SEGMENT=CAR_CSV, SEGTYPE=S0, $
    FIELDNAME=COUNTRY, ALIAS=COUNTRY, USAGE=A7, ACTUAL=A7BV, $
    FIELDNAME=CAR, ALIAS=CAR, USAGE=A10, ACTUAL=A10BV, $
    FIELDNAME=MODEL, ALIAS=MODEL, USAGE=A23, ACTUAL=A23BV, $
    FIELDNAME=BODYTYPE, ALIAS=BODYTYPE, USAGE=A11, ACTUAL=A11BV, $
    FIELDNAME=SEATS, ALIAS=SEATS, USAGE=I1, ACTUAL=A1, $
    FIELDNAME=DEALER_COST, ALIAS=DEALER_COST, USAGE=I5, ACTUAL=A5V, $
    FIELDNAME=RETAIL_COST, ALIAS=RETAIL_COST, USAGE=I5, ACTUAL=A5V, $
    FIELDNAME=SALES, ALIAS=SALES, USAGE=I5, ACTUAL=A5V, $
```

The Delimiter, Enclosure, and Header entries are reflected in the Access File.

Generated Access File:

```
SEGNAME=CAR_CSV, DELIMITER=',', ENCLOSURE=", HEADER=YES, $
```

Reference: Usage and Actual Formats for Delimiters

Prior to Version 7 Release 6.1, server delimiter characters were identified by a field description in the Master File instead of by the DELIMITER parameter of the Access File.

Therefore, older synonyms may use this syntax with a FIELDNAME of DELIMITER, an ALIAS of the delimiter itself, and with USAGE and ACTUAL values as described in the following chart.

For example:

```
FIELDNAME=DELIMITER, ALIAS=',', USAGE=A1, ACTUAL=A1, $
```

USAGE and ACTUAL formats are determined as follows:

Type of delimiter	USAGE	ACTUAL
Printable characters	A*n* where *n* is the number of characters	A*n* where *n* is the number of characters
Non-printable character such as Tab	I4	I1
Group (combination of printable and non-printable characters, or multiple non-printable characters)	Sum of the individual USAGE lengths	Sum of the individual ACTUAL lengths

Reference: Updating Delimited Files

You can update Delimited Flat files using the MODIFY facility. The following delimiter characters are respected:

TAB

A tab character. This is the default.

a

A character string, for example ~.

0x*nn*

A hex code. For example, 0x44 (a comma), or 0x0D0A (a return and a linefeed). The hex code uses ASCII for Windows or UNIX systems and EBCDIC for IBM Mainframes.

Delimited files have been added as a target type for DataMigrator flows.

Chapter 28

Using the Adapter for Git

The adapter for Git allows a vast range of workflow possibilities for projects and teams. The two most commonly used paradigms are Centralized Workflow and Integration-manager Workflow.

In this chapter:

Introducing the Adapter for Git

Git is a free, open source Distributed Version Control System (DVCS), that allows multiple developers to work on the same remote repository, while having a full copy of the remote repository locally.

Preparing the Git Environment

The following jar files are needed to use the adapter for Git. All of the files should be downloaded and saved to your path, since you will be using them during the configuration process in the IBI_CLASSPATH box. The list below includes the name of the file, along with the direct link to its location:

slf4j-api-1.7.25.jar:

https://mvnrepository.com/artifact/org.slf4j/slf4j-api/1.7.25

slf4j-simple-1.6.2.jar:

https://mvnrepository.com/artifact/org.slf4j/slf4j-simple/1.6.2

bcpg-jdk15on-1.60.jar:

https://mvnrepository.com/artifact/org.bouncycastle/bcpg-jdk15on/1.60

bcprov-jdk15on-1.60.jar:

https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk15on/1.60

jsch-0.1.54.jar:

https://mvnrepository.com/artifact/com.jcraft/jsch/0.1.54

org.eclipse.jgit-5.3.0.201903130848-r.jar:

https://mvnrepository.com/artifact/org.eclipse.jgit/org.eclipse.jgit/5.3.0.201903130848-r

commons-io-2.6.jar:

https://mvnrepository.com/artifact/commons-io/commons-io/2.6

Configuring the Adapter for Git

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Adapter for Git

You can configure the adapter from either the Web Console or the Data Management Console.

Note: On Windows, the Environmental Variable PATH should include the directory where the Git utilities are installed. For example, C:\Program Files\Git\usr\bin.

1. From the Web Console Applications page, click *Get Data*.

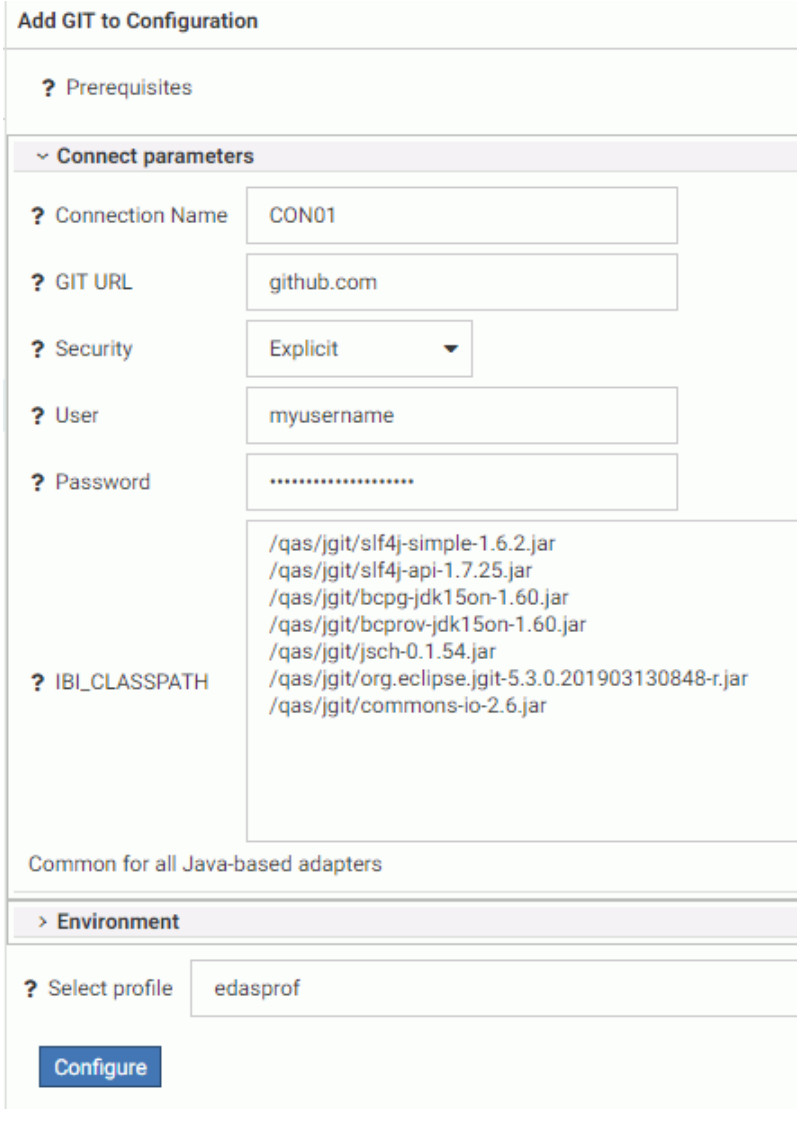
   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click *Git* and click *Configure*.
5. Enter the connection name that is used in the WebFOCUS application.
6. In the GIT URL box, type the URL used to connect to your Git repository.
7. Select *Explicit* as the security type and type your user ID and password.
8. In the IBI_CLASSPATH box, enter the jar file names using the following format: *full_path_GIT_jar_names*.

The following image shows an example of the configuration settings used:



9. Select *edasprof* from the Select profile drop-down menu to add this connection for all users, or select a user profile.
10. Click *Configure*.
11. Click *Test* once the initial configuration is complete. You should see a list of data sources on your server.

To start using the adapter, you must either use the INIT command to create a new repository, or use the CLONE command on an existing one.

Reference: Adapter for Git Configuration Settings

The Adapter for Git is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Git URL

The full computer name or URL for the site where the remote repositories are located.

Note: A remote repository can be located on a remote network location or on your local file system.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

The user ID used to connect to the Git server.

Password

The password used to connect to the Git server.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. This value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions when setting values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar). When editing the file manually, you must maintain the colon delimiter.

For the Git adapter, the full path GIT jar file names are entered.

Select profile

Select a profile from the drop-down menu to store the configured connection.

Chapter 29

Using the Adapter for Google Analytics

This section describes how to configure the Google Analytics Adapter.

In this chapter:

Google Analytics Adapter Overview

The Google Analytics Adapter is used to report against the information residing in the Google Analytics environment. Metrics, such as Page Views and Users, can be analyzed by various dimensions (for example, Country and City).

You can configure the Google Analytics Adapter using the WebFOCUS Reporting Server Web Console. The adapter requires a connection, which stores the access token. A valid Google Analytics access token is required to issue Google Analytics API calls. This token is associated with a Google Analytics application and a specific Google Analytics user.

Creating a Google Project

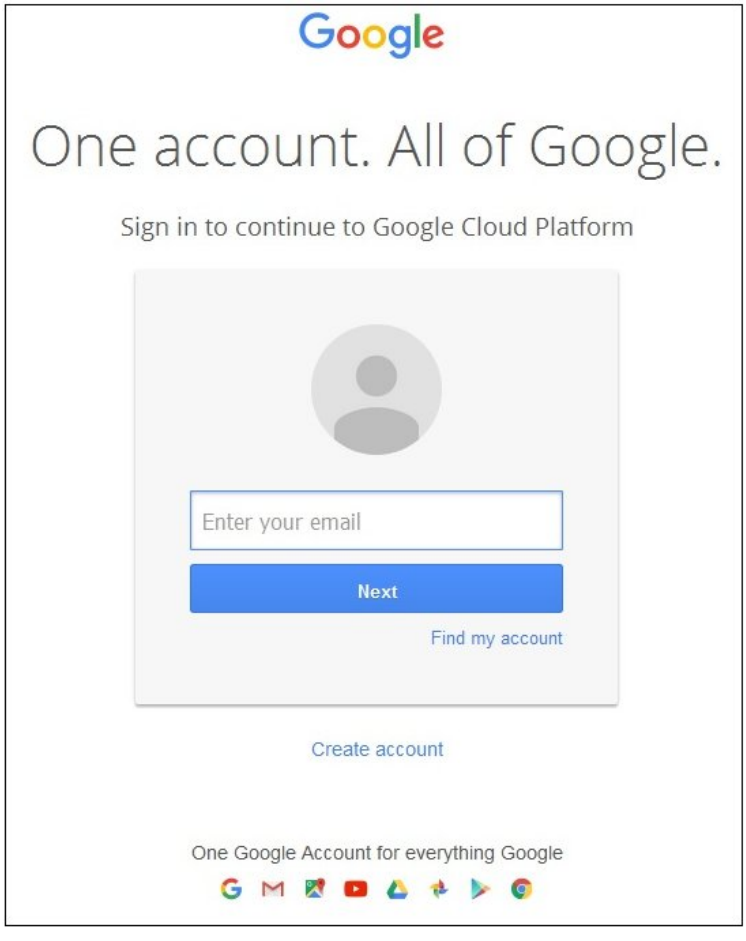
A Google project must be available before you can configure the Google Analytics Adapter.

Procedure: How to Create a Google Project

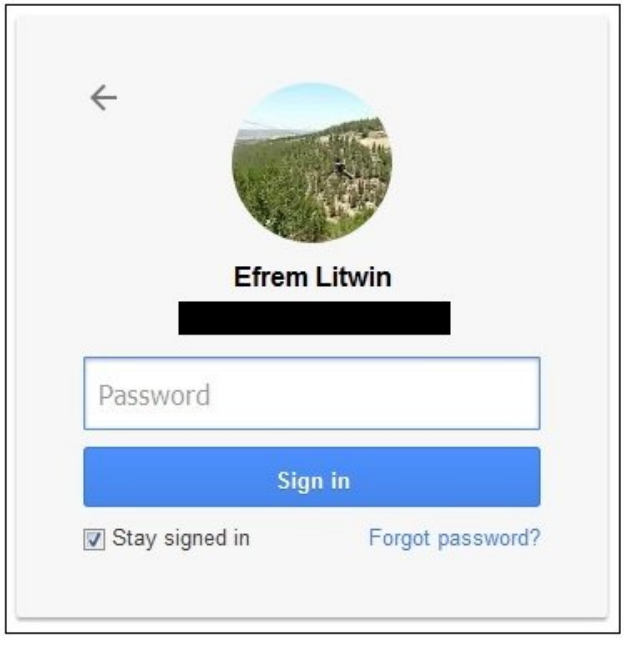
1. Enter the following URL in a web browser:

   `https://console.developers.google.com/project`

If you are not already signed into Google, a Sign in dialog for the Google Cloud Platform opens, as shown in the following image.

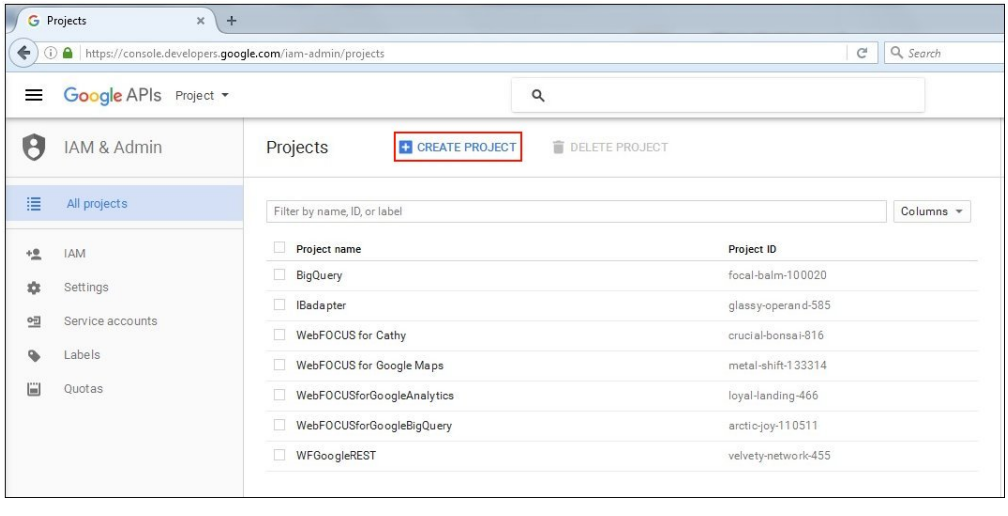


2. Enter an email address for the Google account that has administrative rights to the Google Analytics environment, and then click *Next*.



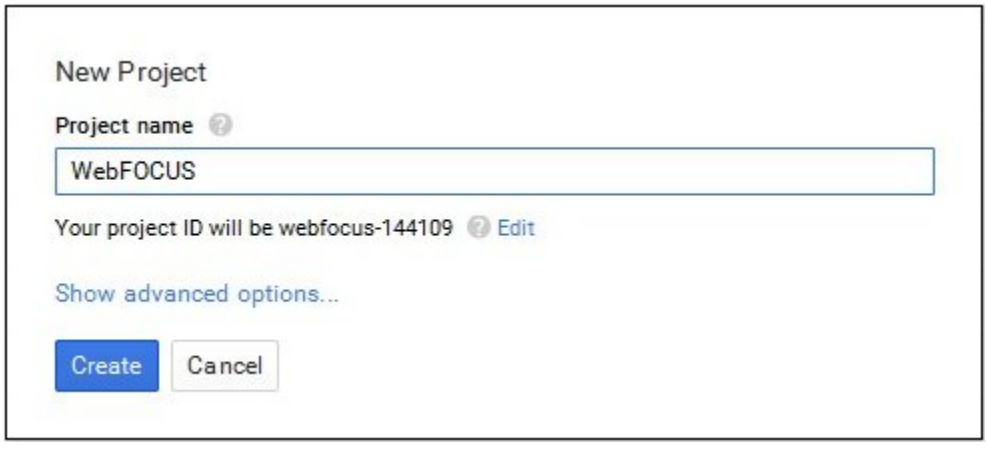
3. Enter a valid password for the Google account, and then click *Sign In*.

   The Projects screen opens, as shown in the following image.



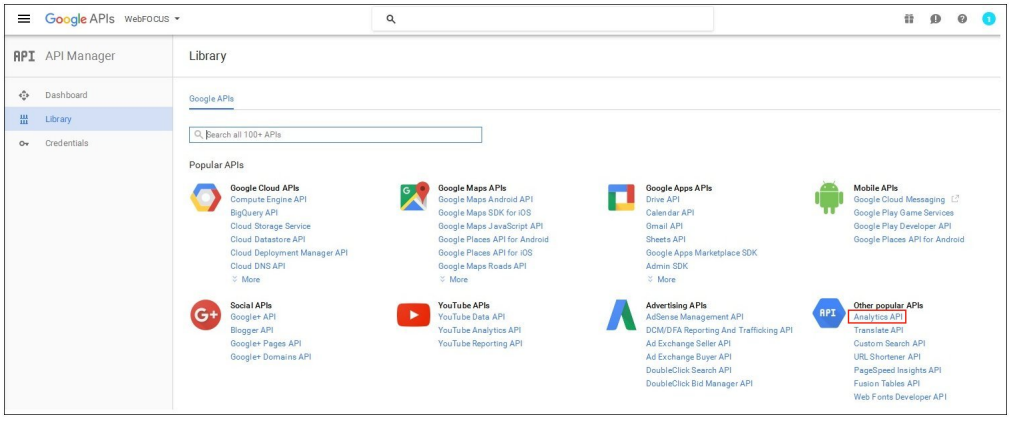
4. Click *CREATE PROJECT*.

The New Project screen opens, as shown in the following image.



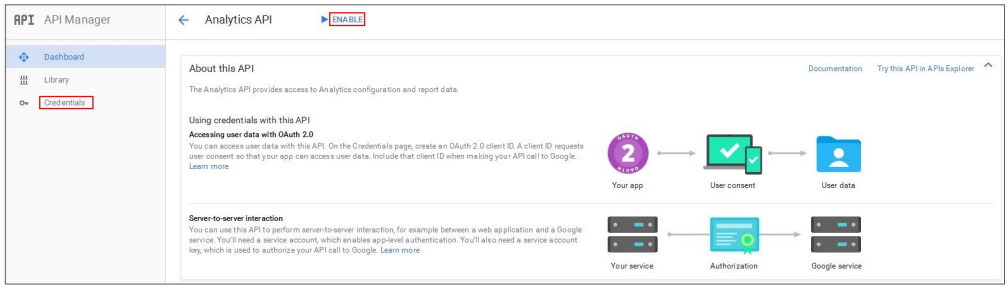
5. Enter a name for your new project and then click *Create*.

   The Library screen opens, as shown in the following image.



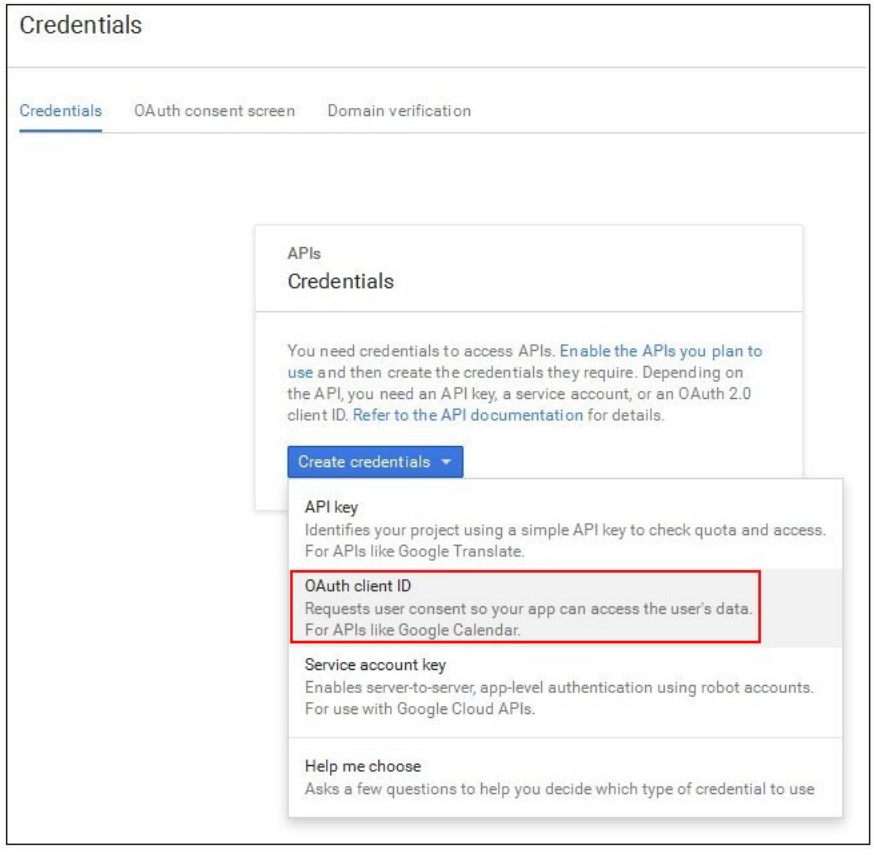
6. Click *Analytics API* under the *Other popular APIs* group.

The Analytics API screen opens, as shown in the following image.



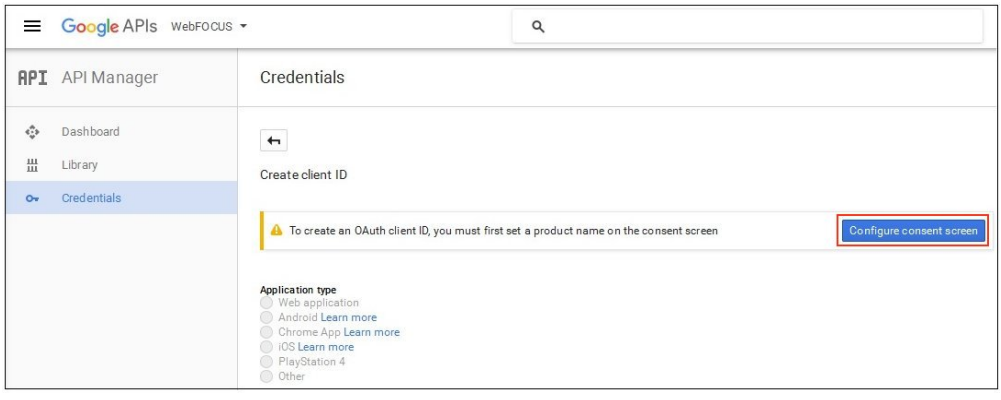
7. Click *ENABLE*.
8. Click *Credentials* in the left pane.

   The Credentials screen opens, as shown in the following image.



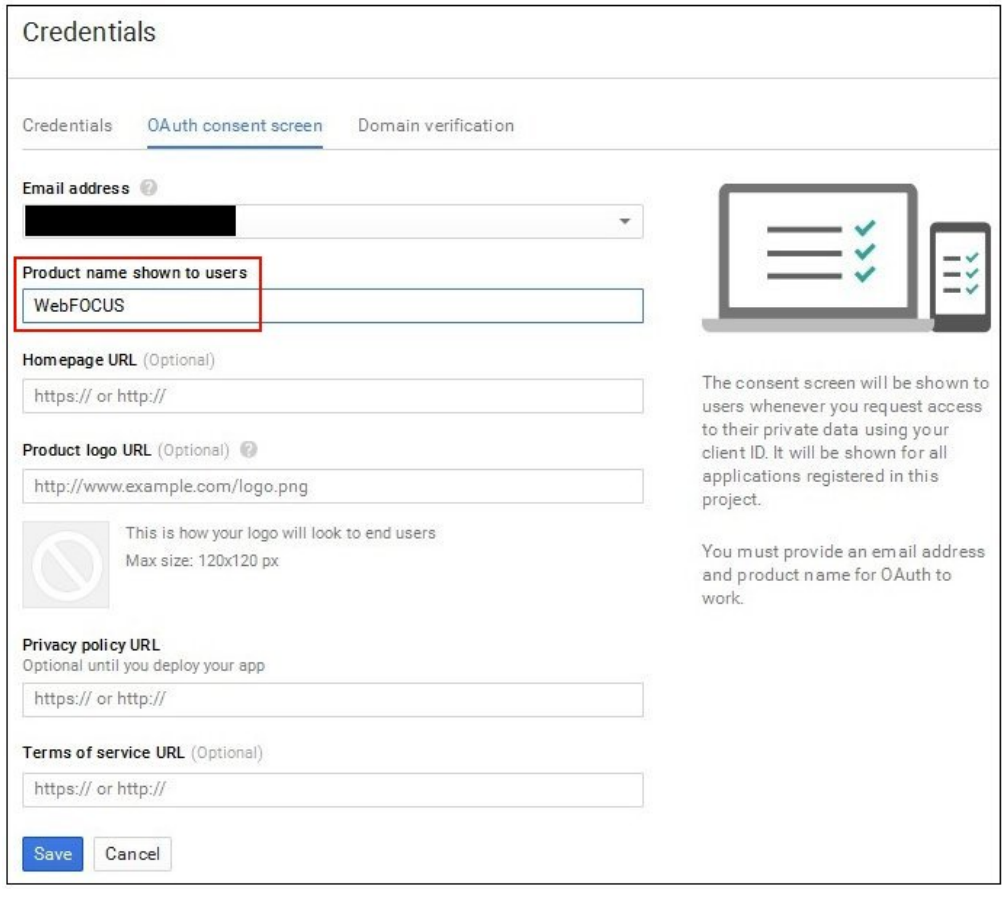
9. Click the *Create credentials* drop-down list and select *OAuth client ID*.

   The Create client ID screen opens, as shown in the following image.



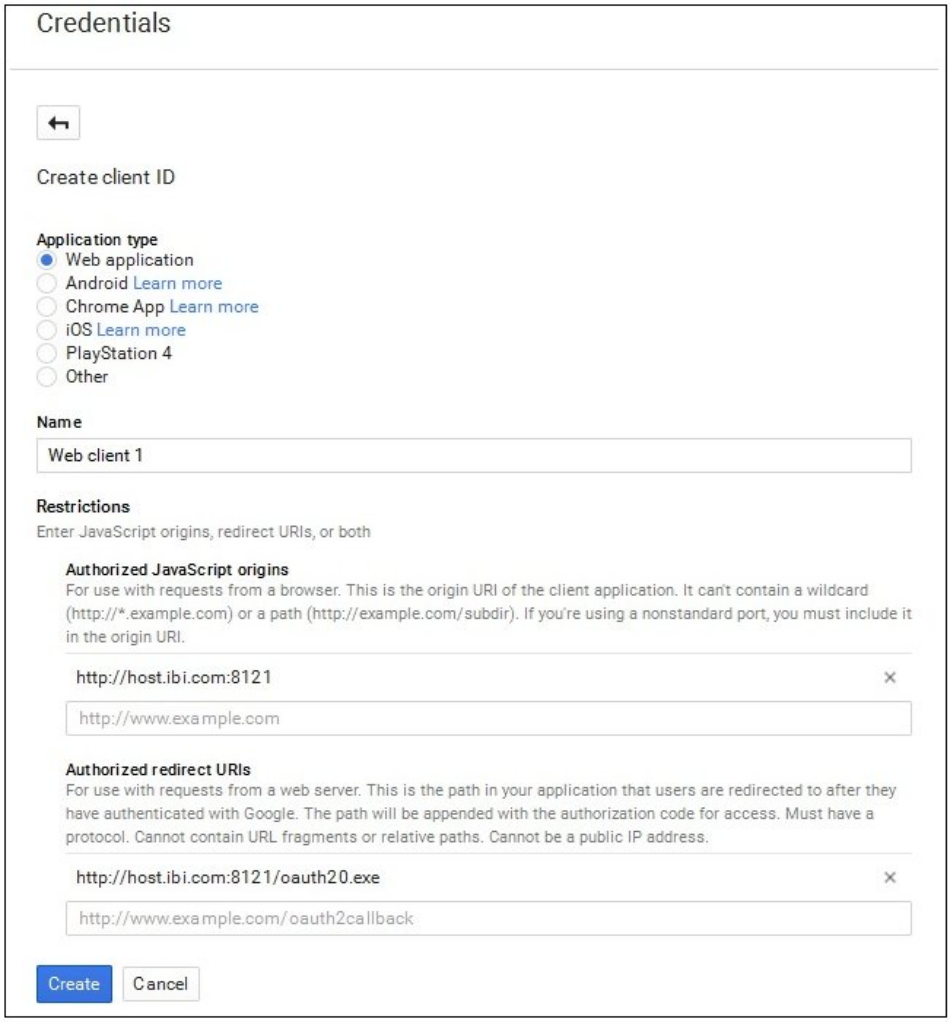
10. Click *Configure consent screen*.

The OAuth consent screen opens, as shown in the following image.



11. Enter a value in the *Product name shown to users* field and click *Save*.

The Application type screen opens, as shown in the following image.



12. Perform the following steps:
    a. Select *Web application* from the list of application types.
    b. Enter the host name and port used to access the WebFOCUS Reporting Server Web Console in the *Authorized JavaScript origins* field.

       For example:

       `http://host.ibi.com:8121`

If the WebFOCUS Reporting Server is installed as a standalone server, then `http://localhost` should be specified as the value in the *Authorized JavaScript origins* field.

c. Enter the host name and port used to access the WebFOCUS Reporting Server Web Console with oauth20.exe in the *Authorized redirect URIs* field.

For example:

```
http://host.ibi.com:8121/oauth20.exe
```

If the WebFOCUS Reporting Server is installed as a standalone server, then `http://localhost/oauth20.exe` should be specified as the value in the *Authorized redirect URIs* field.

13. Click *Create*.

The OAuth client screen opens, which displays your *client ID* and *client secret* values, as shown in the following image.

The *client ID* and *client secret* values are required to configure the Google Analytics Adapter.

14. Click *OK*.

You are now ready to obtain the Web Profile ID for a website within the Google Analytics environment.

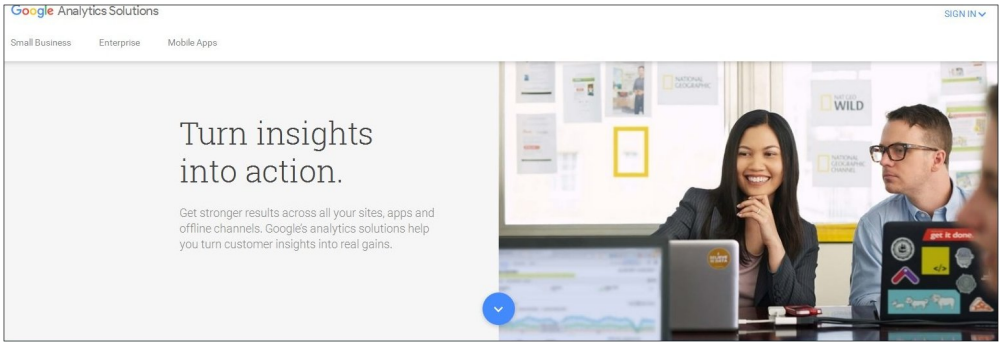
Obtaining the Web Profile ID

This section describes how to obtain the Web Profile ID for a website within the Google Analytics environment. The Web Profile ID is required to configure the Google Analytics Adapter.

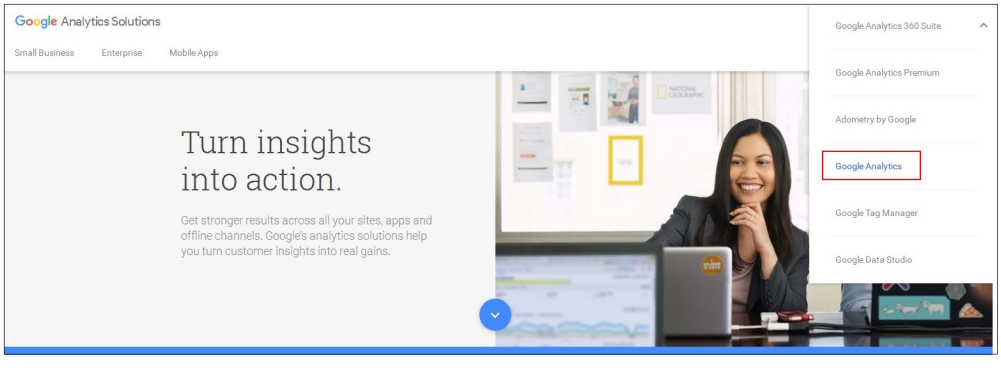
Procedure: How to Obtain the Web Profile ID

1. Enter the following URL in a web browser:

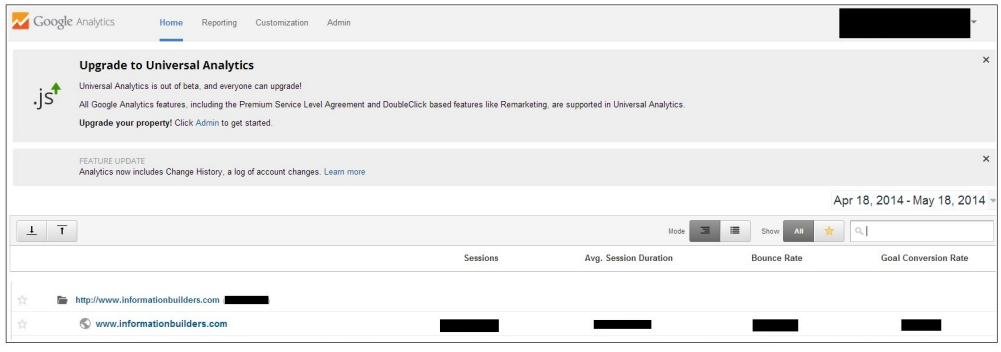
   `https://www.google.com/analytics`



2. Click *SIGN IN* in the upper-right corner of the page.
3. Select *Google Analytics*, as shown in the following image.

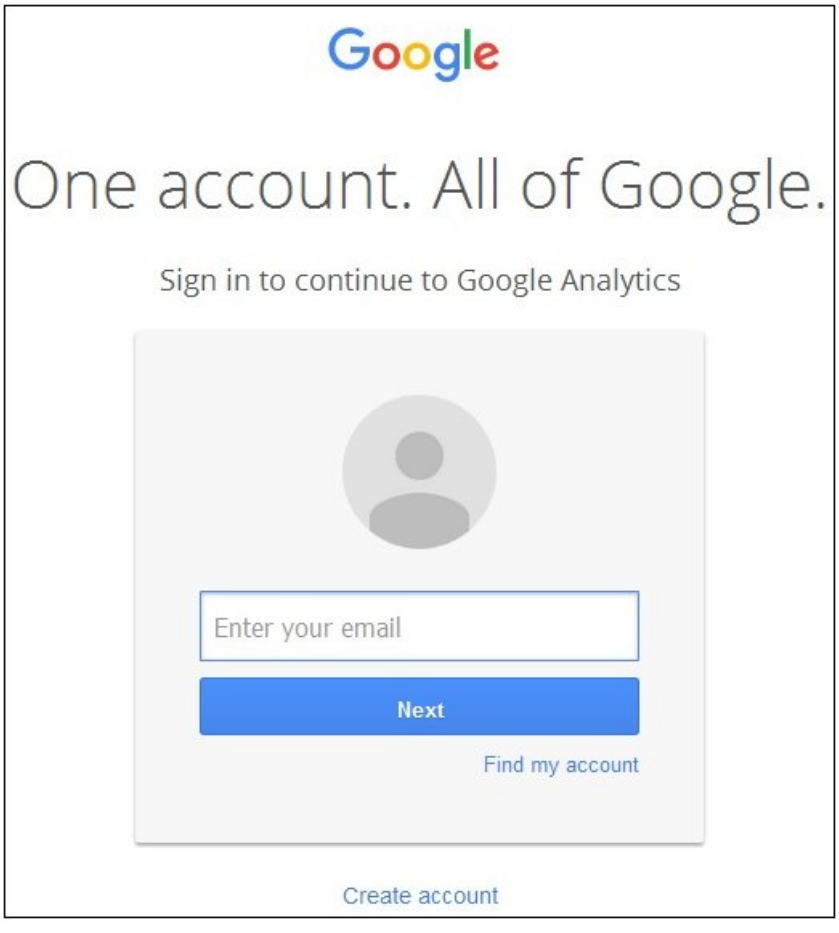


If you are already signed in to Google, then the Google Analytics page which lists the configured websites opens, as shown in the following image.

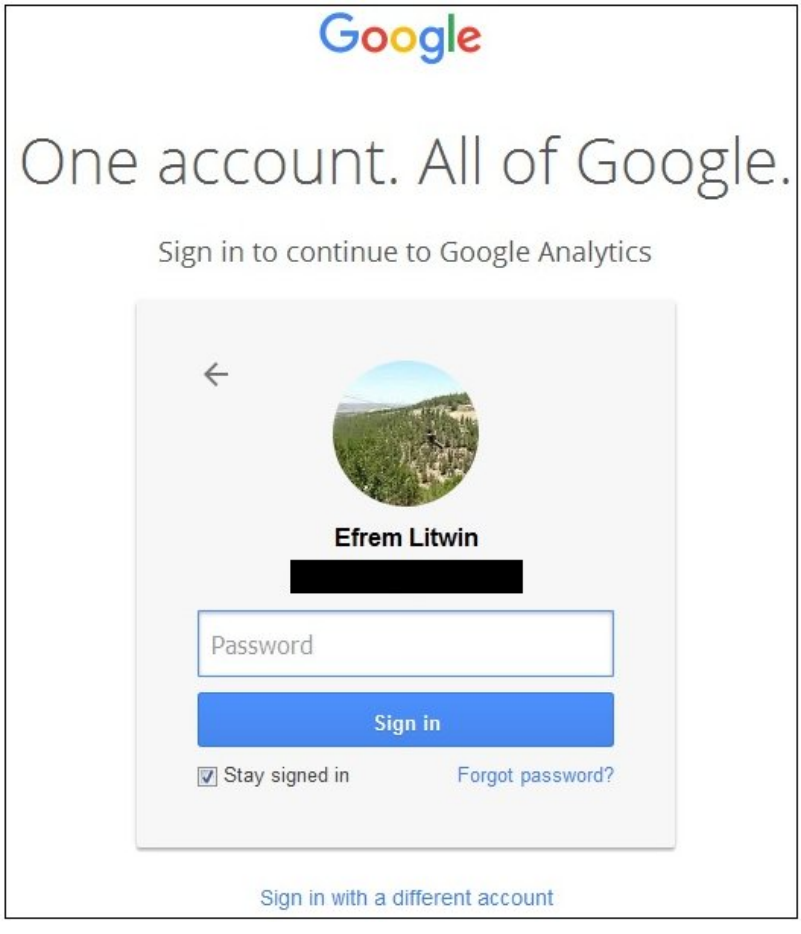


If you are not already signed in to Google Analytics, then click the *SIGN IN* link in the upper-right corner of the page.

A Sign in to Google Analytics page opens, as shown in the following image.



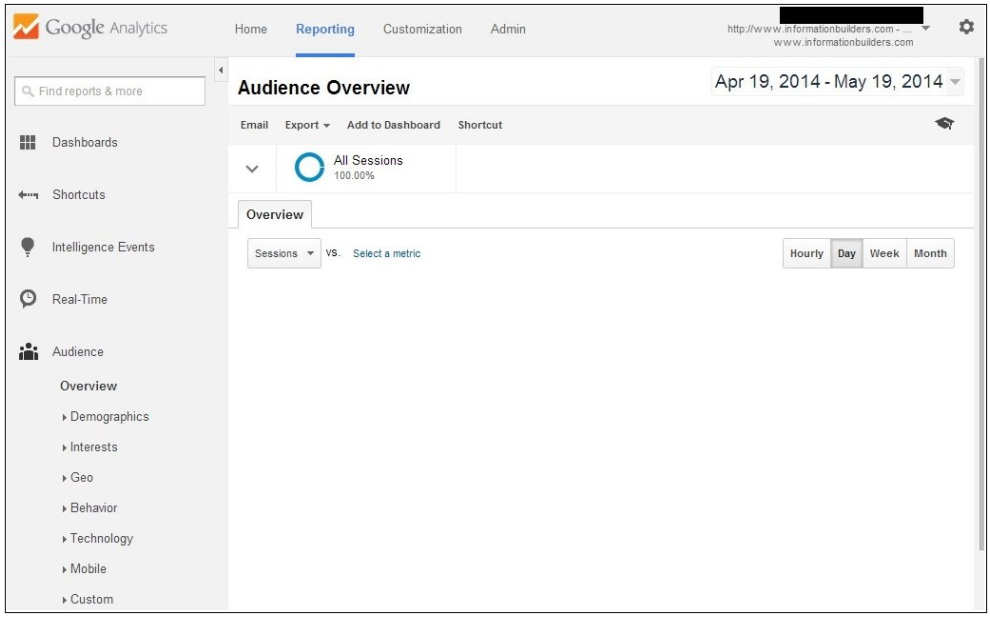
Enter an email address for the Google account that has administrative rights to the Google Analytics environment, and then click *Next*.



Enter a valid password for the Google account, and then click *Sign In*.

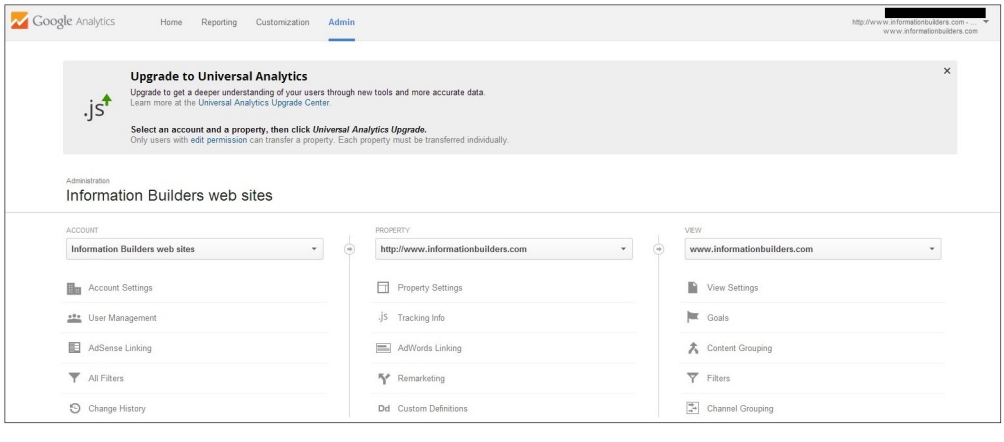
4. Click the link to the website that will be used during the configuration of the Google Analytics Adapter.

The Google Analytics Reporting page for the selected website opens, as shown in the following image.



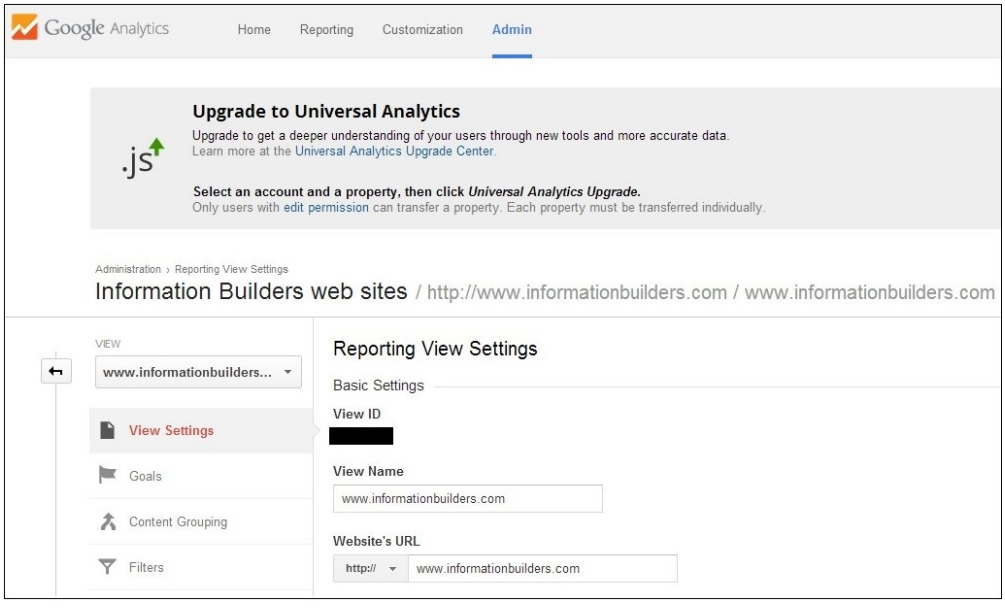
5. Click the *Admin* link at the top of the page.

   The Google Analytics Admin page for the selected website opens, as shown in the following image.



6. Click the *View Settings* link, which is located in the View column.

The Google Analytics Reporting View Settings page for the selected website opens, as shown in the following image.



The View ID value is required to configure the Google Analytics Adapter.

Configuring the Google Analytics Adapter

This section describes how to configure the Google Analytics Adapter.

Procedure: How to Configure the Google Analytics Adapter

1. Clear the cookies from the web browser that will be used to start the WebFOCUS Reporting Server Web Console.
2. Access the WebFOCUS Reporting Server Web Console using the host name and port that you specified in the AUTHORIZED JAVASCRIPT ORIGINS field of the Google project.

   For example:

   `http://host.ibi.com:8121`

   For more information, see *How to Create a Google Project* on page 777.
3.
4. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

5. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
6. Right-click the *Google Analytics* node and select *Configure*.

   The Add Google Analytics to Configuration pane opens, as shown in the following image.

7. Enter the values for the Client ID and Client Secret as defined by the Client ID and Client secret respectively in the Google project.

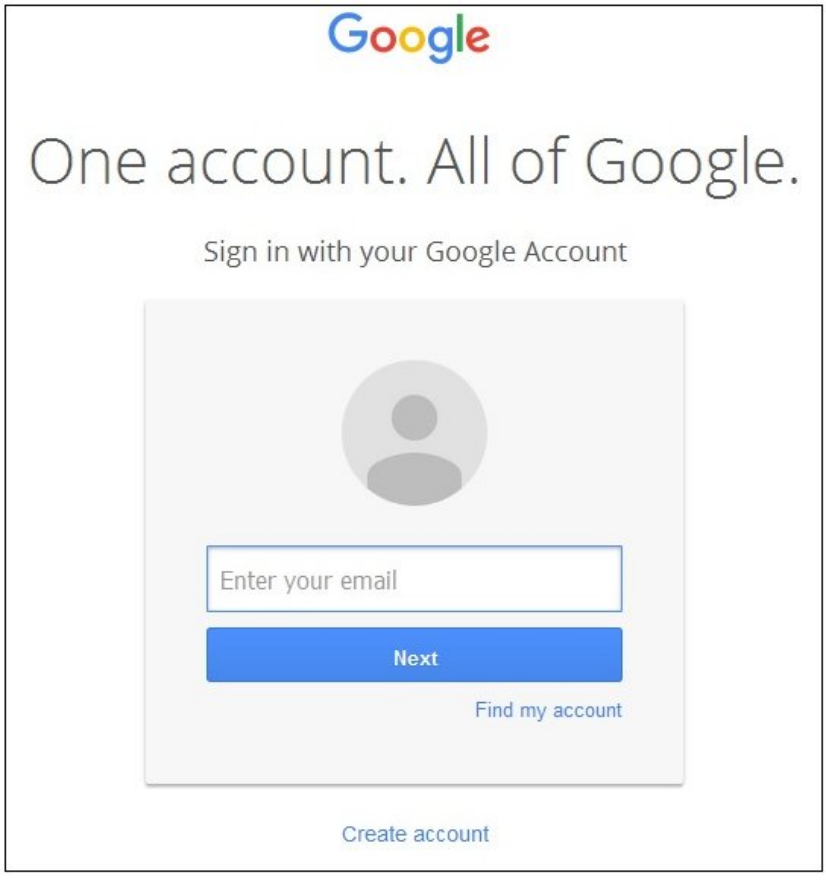
   For more information, see *How to Create a Google Project* on page 777.
8. Enter the value for the Web Profile ID as defined by the View ID in the Google Analytics Reporting View Settings for the selected website.

   This value is prefixed by *ga*. For example:

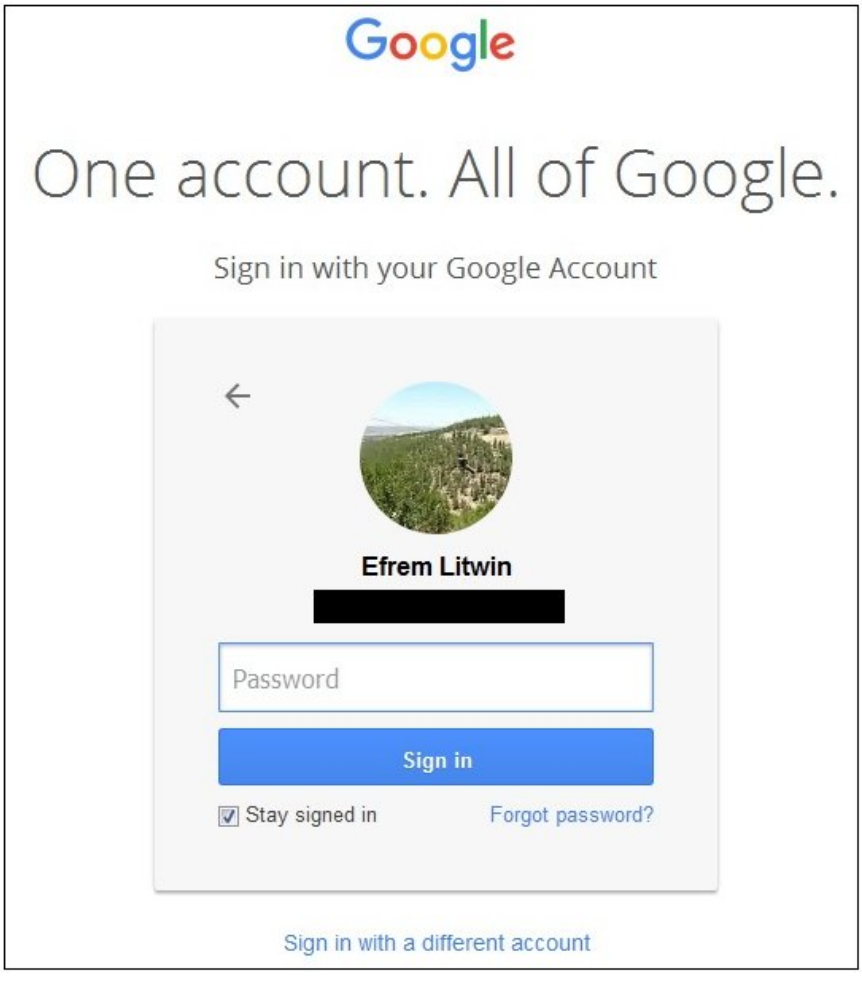
   ```
   ga:87878787
   ```

   For more information, see *How to Obtain the Web Profile ID* on page 786.
9. Click the *Get Access Token* link.

A Google Sign In page opens, as shown in the following image.

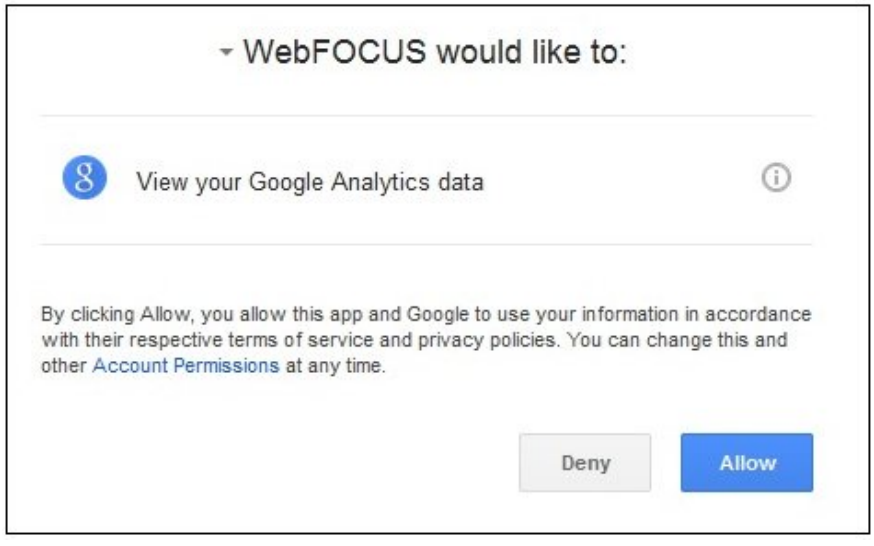


10. Enter an email address for the Google account that has administrative rights to the Google Analytics environment, and then click *Next*.



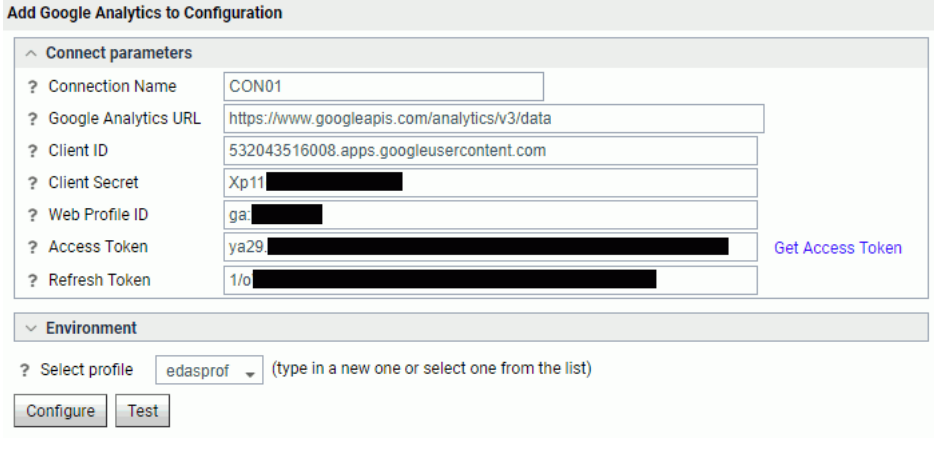
11. Enter a valid password for the Google account, and then click *Sign In*.

The View your Google Analytics data consent screen opens, as shown in the following image.



12. Click *Allow*.

    You are returned to the Add Google Analytics to Configuration pane, where the Access Token field and Refresh Token field are now populated, as shown in the following image.



13. Click *Configure*.

The Google Analytics Adapter is added to the configured Adapters list in the navigation pane.

Reference: Connection Attributes for Google Analytics

The following list describes the connection attributes for the Google Analytics Adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Google Analytics URL

The URL of the Google Analytics API request. The default value is:

```
https://www.googleapis.com/analytics/v3/data
```

Client ID

The value that identifies your application to Google Analytics.

Obtain this value using the following steps:

1. Go to:

   ```
   https://cloud.google.com/console/project
   ```

2. Click on the Project Name for the Google Analytics Adapter application that was previously created.
3. Click *APIs & auth* in the left pane.
4. Click *Credentials* in the left pane.
5. Use Client ID in the Client ID for web application section.

Client Secret

The value which identifies your application to Google Analytics. This value is used in conjunction with Client ID.

Obtain this value using the following steps:

1. Go to:

   ```
   https://cloud.google.com/console/project
   ```

2. Click on the Project Name for the Google Analytics Adapter application that was previously created.
3. Click *APIs & auth* in the left pane.

4. Click *Credentials* in the left pane.
5. Use Client secret in the Client ID for web application section.

Web Profile ID

The ID that identifies the view (profile) for a Google Analytics account.

Obtain this value using the following steps:

1. Go to:

   ```
   http://www.google.com/analytics
   ```

2. Sign in with Google credentials that have administrative rights to Google Analytics.
3. Click on the website that is to be analyzed (for example, www.informationbuilders).
4. Click *Admin* in the upper-right corner of the screen.
5. Click *View Settings*.
6. Look for View ID under Basic Settings.

   This value is prefixed by *ga*. For example:

   ```
   ga:87878787
   ```

Access Token

The value that identifies the user your application is acting on behalf. Click the *Get Access Token* link to obtain this token and the Refresh Token.

In order for the Get Access Token to complete successfully, the host name used to access the WebFOCUS Reporting Server Web Console must match the host name specified for the Redirect URI in the Google Analytics application.

A Google sign-on screen opens if you are not already logged into a Google account.

A Consent screen opens. Click *Allow Access*.

If an issue arises when obtaining the Access and Refresh Tokens, clear your browser cache, including cookies.

Refresh Token

The Access Token has a very short lifespan. The Refresh Token is used to obtain a new Access Token during run time.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the connection attributes. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), select *New Profile* from the drop-down list and enter a name in the Profile Name field (the extension is added automatically).

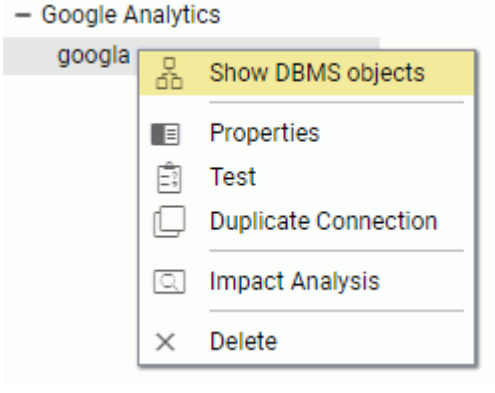
Store the connection attributes in the server profile (edasprof).

Creating Metadata for the Google Analytics Adapter

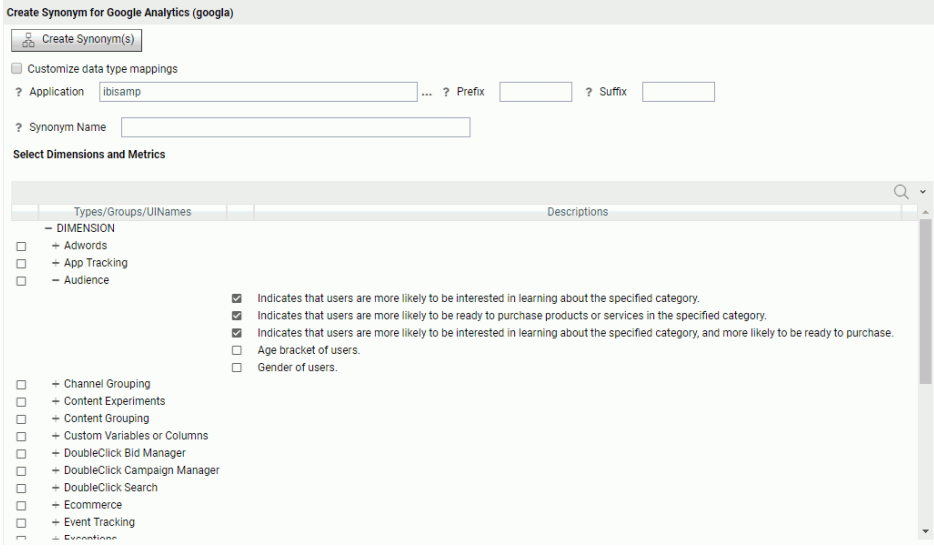
Create Synonym for the Google Analytics Adapter creates the metadata used for WebFOCUS reporting.

Procedure: How to Create Metadata

1. From the WebFOCUS Reporting Server Web Console, expand the *Adapters* folder, *Configured* folder, and then the *Google Analytics* folder.
2. Right-click the configured connection for the Google Analytics Adapter (for example, GoogleAnalytics) and select *Show DBMS objects* from the context menu, as shown in the following image.



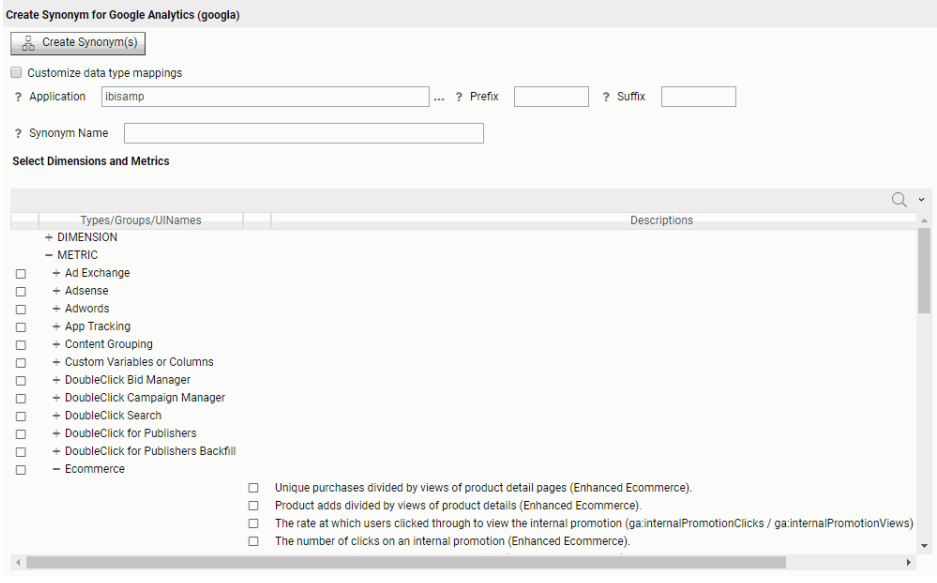
The Create Synonym for Google Analytics pane opens, as shown in the following image.



3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata is to be stored.
4. Enter a Synonym Name that will be used to store the metadata.
5. Expand the *Dimensions* tree and select the dimensions that will be used for analysis.

   Note: Currently, Google Analytics only support 7 dimensions for a synonym. Google might increase this number sometime in the future.

Scroll down within the *Select Dimensions and Metrics* matrix and expand the *Metric* tree, as shown in the following image.



6. Select the metrics that will be used for analysis.

   Note: Currently, Google Analytics only support 10 metrics for a synonym. Google might increase this number sometime in the future.

7. Click *Create Synonym(s) and Examples*.

   The Create Synonym for Google Analytics Status pane opens and indicates that the synonym was created successfully.

Chapter 30

Using the Adapter for Google BigQuery

This section describes how to configure the Google BigQuery Adapter.

In this chapter:

Google BigQuery Adapter Overview

The Google BigQuery Adapter is used to load data into the Google BigQuery environment and report against the information that is residing in the Google BigQuery environment.

You can configure the Google BigQuery Adapter using the WebFOCUS Reporting Server Web Console. The adapter requires a connection, which stores the access token. A valid Google BigQuery access token is required to issue Google BigQuery API calls. This token is associated with a Google BigQuery application and a specific Google BigQuery user.

Creating a Google BigQuery Project

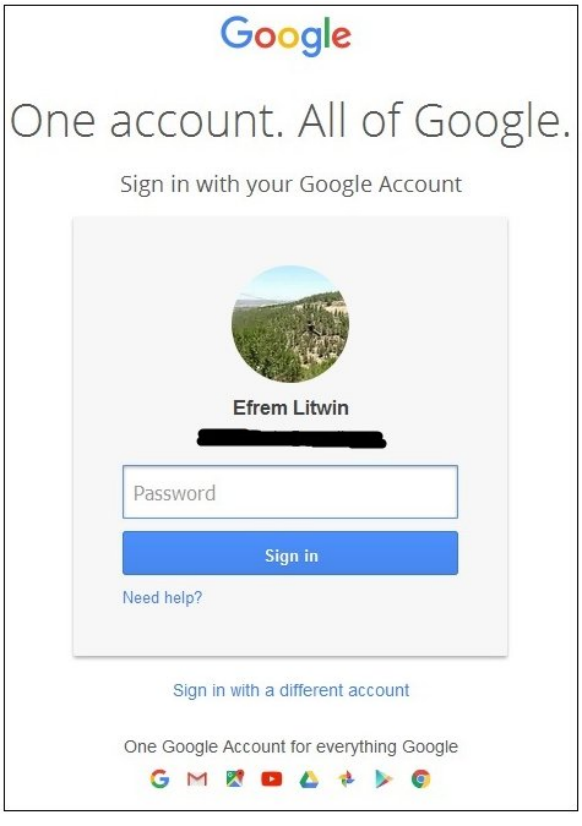
A Google project must be available before you can configure the Google BigQuery Adapter.

Procedure: How to Create a Google Project

1. Enter the following URL in a web browser:

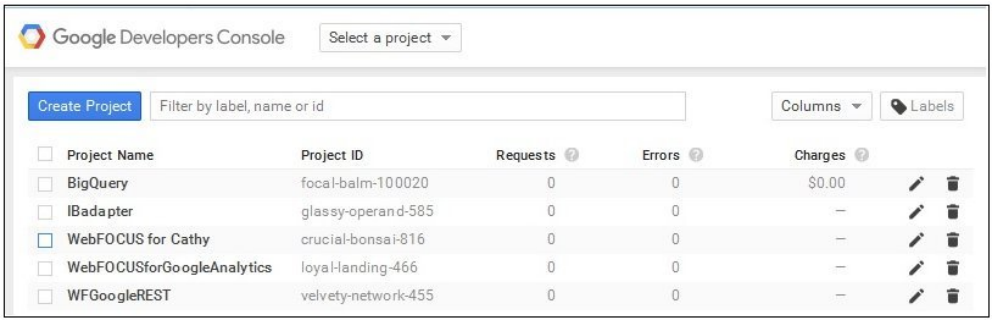
   `https://console.developers.google.com/project`

If you are not already signed into a Google account, a Sign in dialog for the Google Developers Console opens, as shown in the following image.



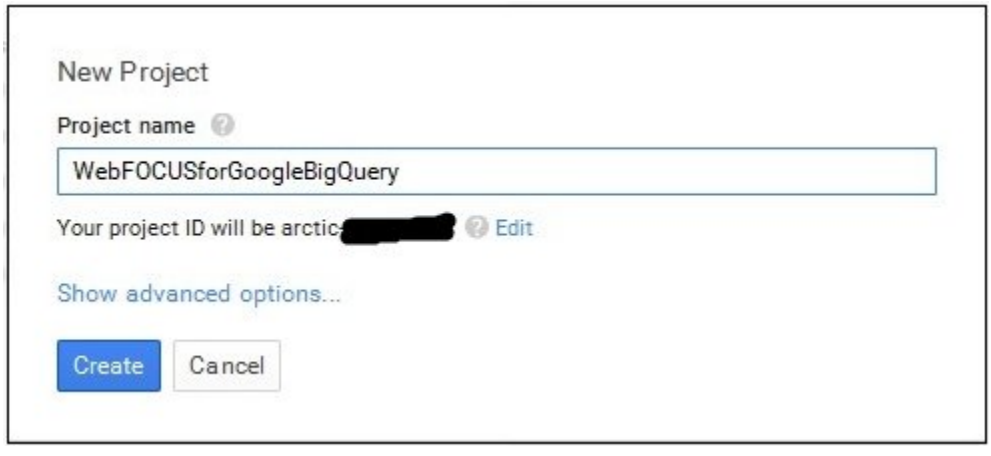
2. Enter the Google sign in credentials using an account that has administrative rights to the Google BigQuery environment, and then click *Sign In*.

   The Google Developers Console screen opens, as shown in the following image.



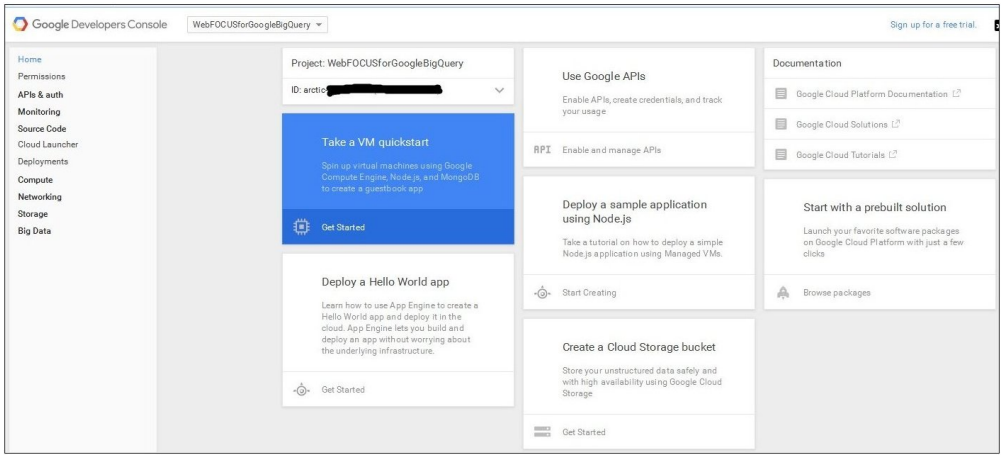
3. Click *CREATE PROJECT*.

   The New Project screen opens, as shown in the following image.



4. Enter a project name and then click *Create*.

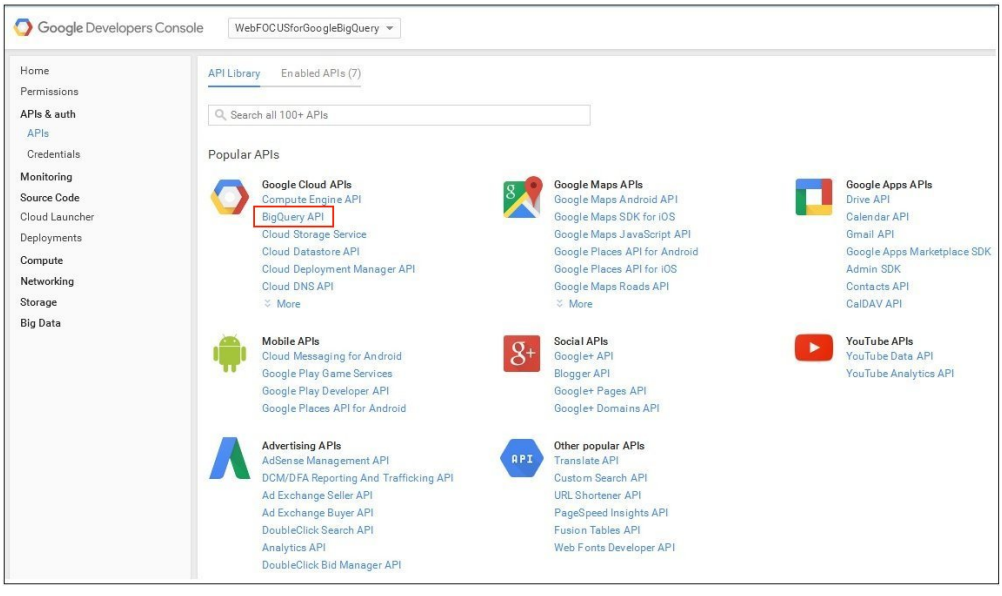
   The page that is used to configure the project opens, as shown in the following image.



   Note: The ID will be used in the Google BigQuery Adapter configuration.

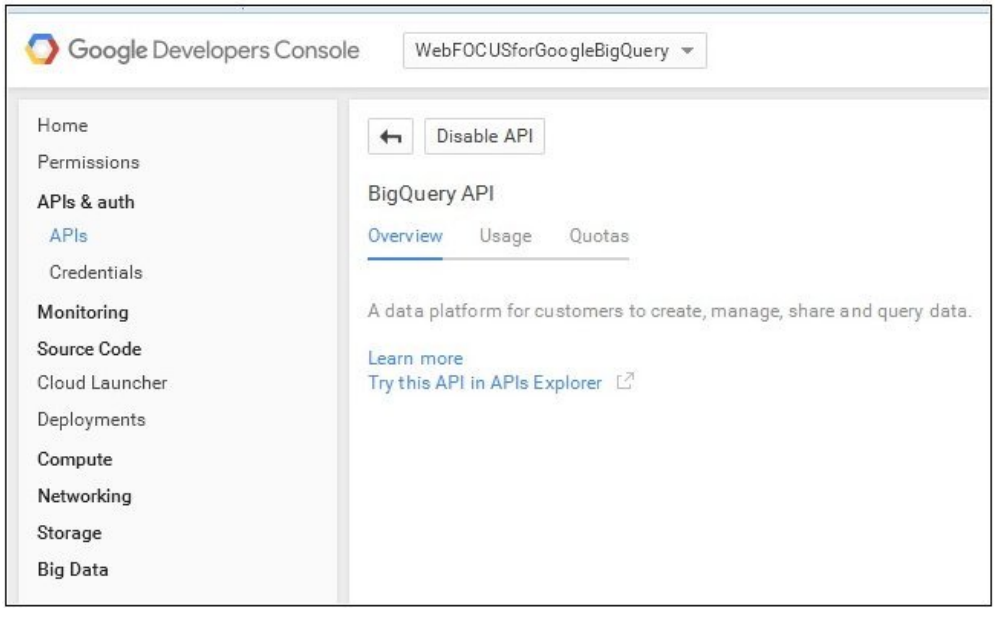
5. Click *APIs & auth* in the left pane.
6. Click *APIs* in the left pane.

The API Library screen opens, as shown in the following image.



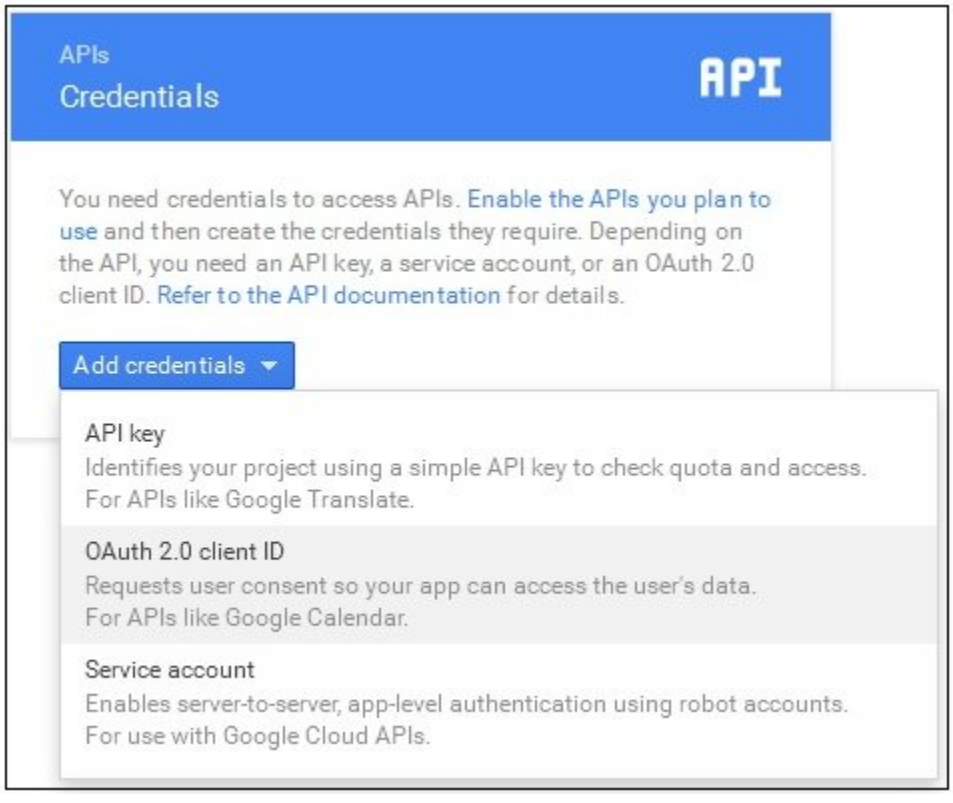
7. Click *BigQuery API*.

The BigQuery API is enabled, as shown in the following image.



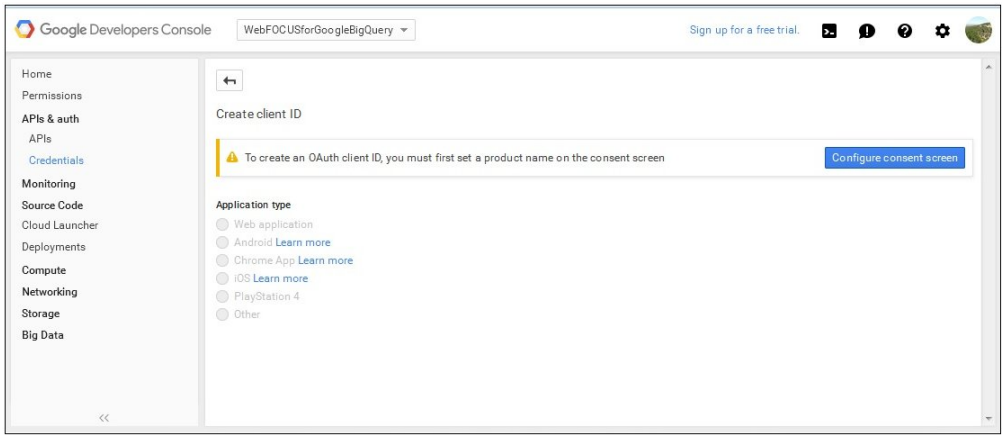
8. Click *Credentials* in the left pane.

   The APIs Credentials screen opens, as shown in the following image.



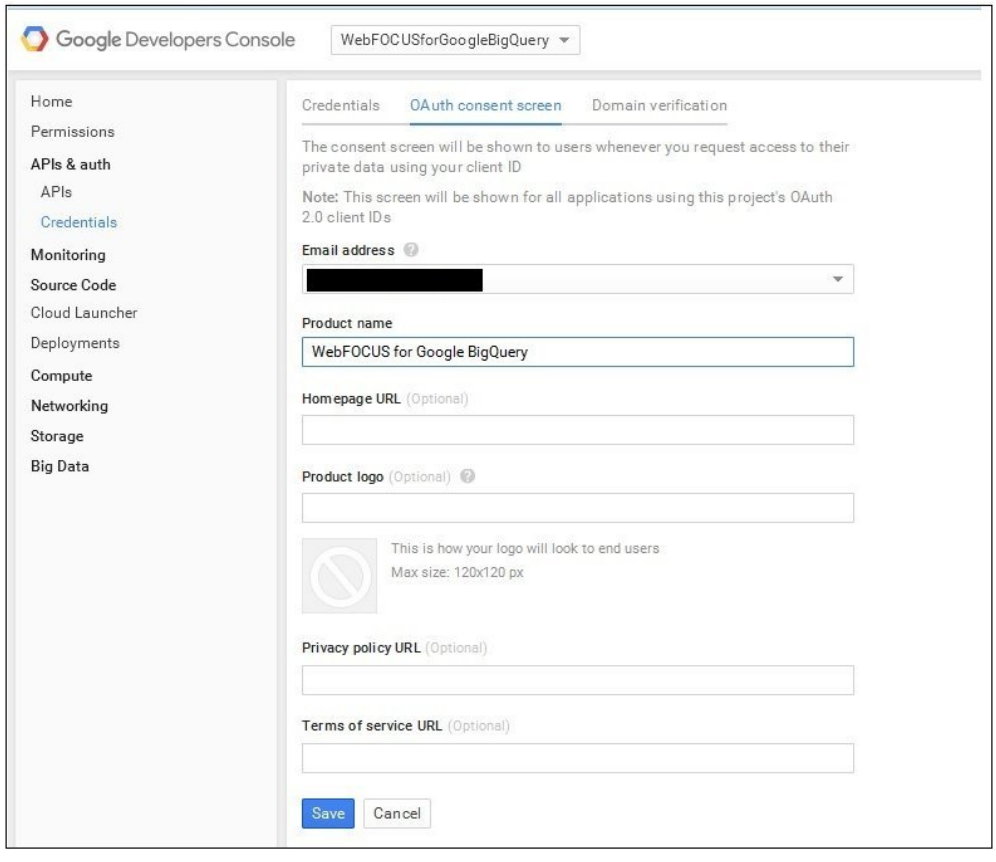
9. From the Add credentials drop-down list, click *OAuth 2.0 client ID*.

The Create client ID screen opens, as shown in the following image.



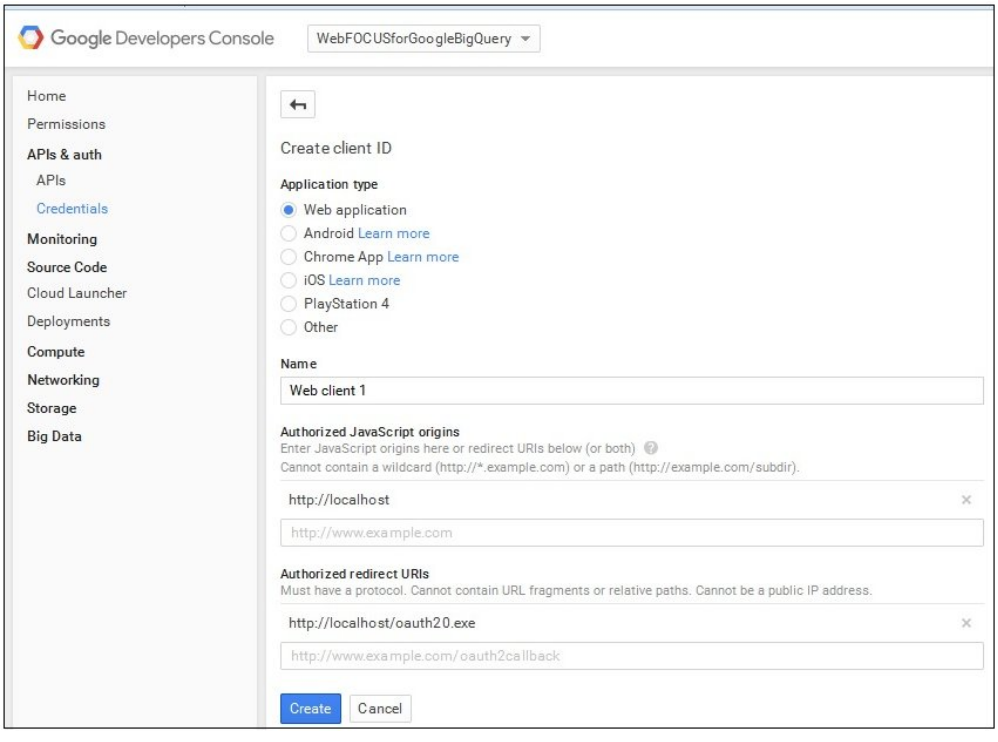
10. Click *Configure consent screen*.

The OAuth consent screen configuration opens, as shown in the following image.



11. Enter the Product Name that will appear on the Consent Screen, which will appear when you configure the Google BigQuery Adapter.
12. Click *Save*.

The Create Client ID page opens, as shown in the following image.



13. Perform the following steps:

    a. Select *Web application* from the list of application types.

    b. Enter the host name and port used to access the WebFOCUS Reporting Server Web Console in the AUTHORIZED JAVASCRIPT ORIGINS field.

       For example:

       ```
       http://host.ibi.com:8121
       ```

       If the WebFOCUS Reporting Server is installed as a standalone server, then *http://localhost* should be specified as the value in the AUTHORIZED JAVASCRIPT ORIGINS field.

    c. Enter the host name and port used to access the WebFOCUS Reporting Server Web Console with oauth20.exe in the AUTHORIZED REDIRECT URI field.

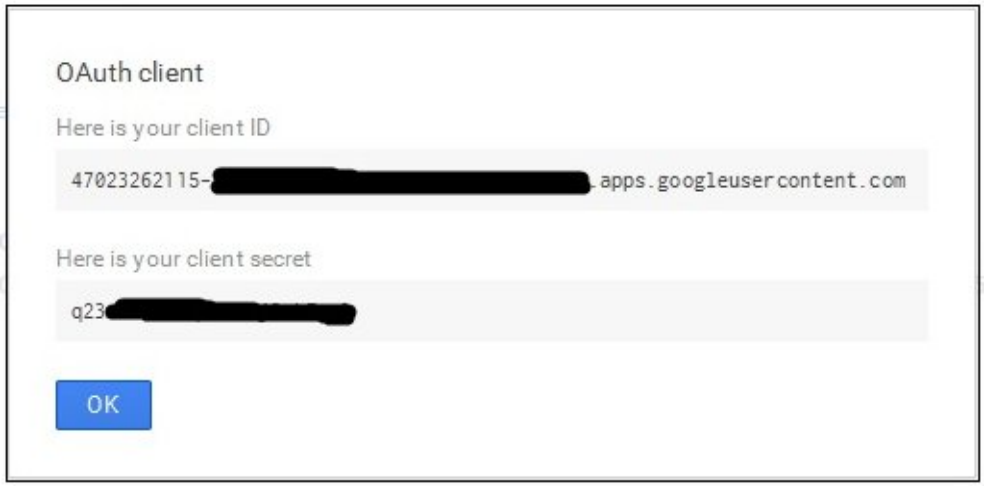
       For example:

       ```
       http://host.ibi.com:8121/oauth20.exe
       ```

If the WebFOCUS Reporting Server is installed as a standalone server, then *http://localhost/oauth20.exe* should be specified as the value in the AUTHORIZED REDIRECT URI field.

14. Click *Create*.

    The OAuth client screen containing the Client ID and Client Secret opens, as shown in the following image.



    Note: The *Client ID* and *Client secret* values are required to configure the Google BigQuery Adapter.

15. Click *OK*.

Configuring the Google BigQuery Adapter

This section describes how to configure the Google BigQuery Adapter.

Procedure: How to Configure the Google BigQuery Adapter

1. Clear the cookies from the web browser that will be used to start the WebFOCUS Reporting Server Web Console.
2. Access the WebFOCUS Reporting Server Web Console using the host name and port that you specified in the AUTHORIZED JAVASCRIPT ORIGINS field of the Google project.

   For example:

   ```
   http://host.ibi.com:8121
   ```

   For more information, see *How to Create a Google Project* on page 801.

3. From the Web Console Applications page, click *Get Data*.

   or

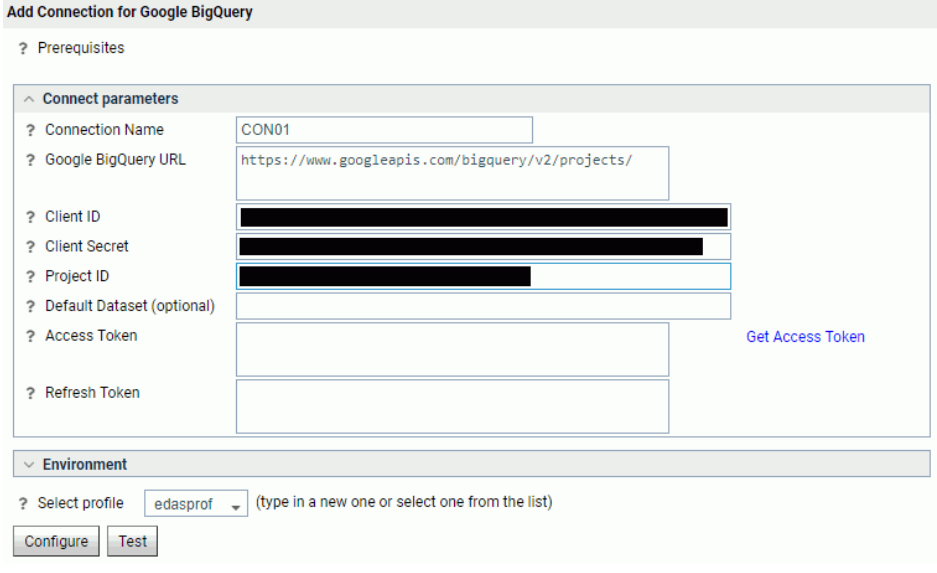
   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

4. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

5. Right-click the *Google BigQuery* node and select *Configure*

   The Add Connection for Google BigQuery pane opens, as shown in the following image.

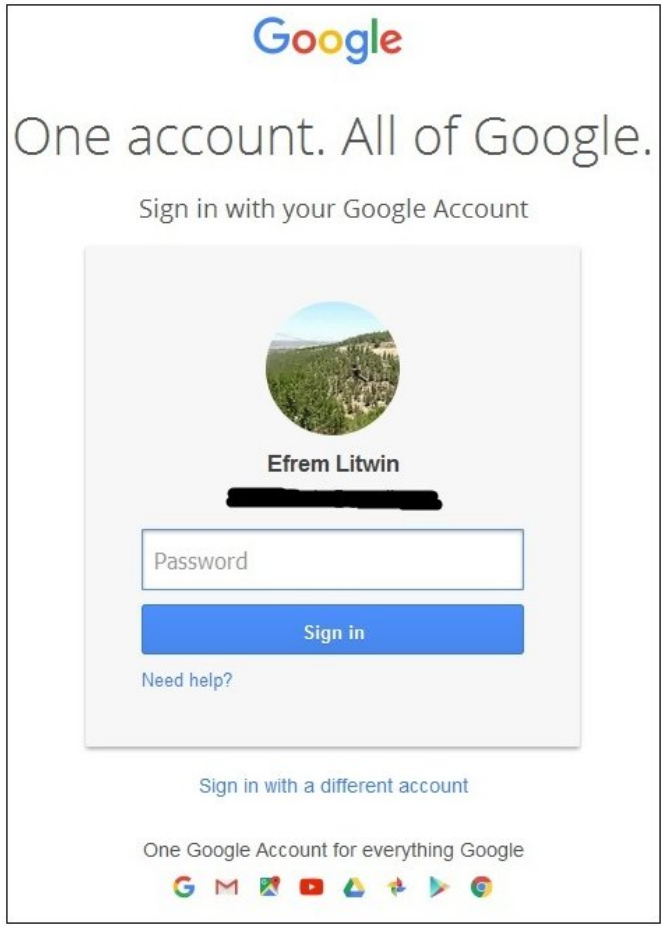


6. Enter the values for the Client ID, Client Secret, and Project ID as defined by the Client ID, Client secret, and Project ID respectively in the Google project.

   For more information, see *How to Create a Google Project* on page 801.

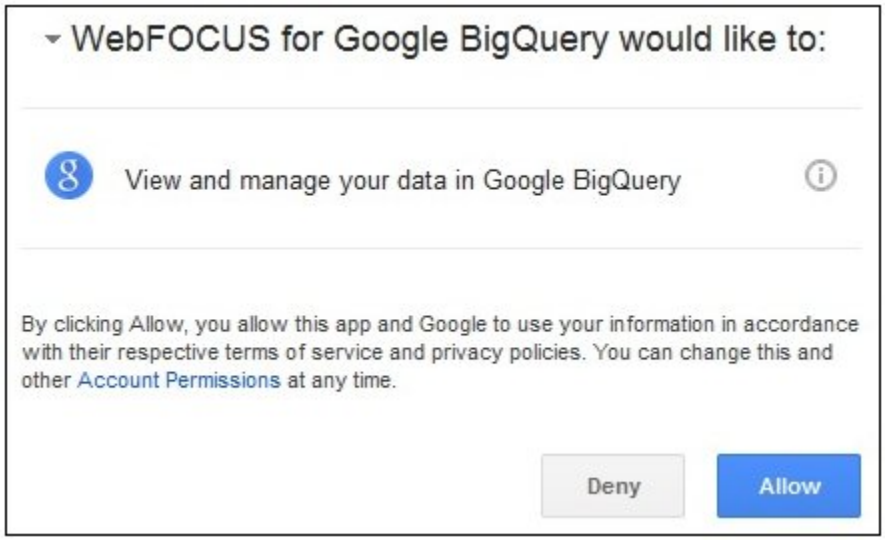
7. Click the *Get Access Token* link.

If you are not already signed into a Google account, a Google Sign In page opens, as shown in the following image.



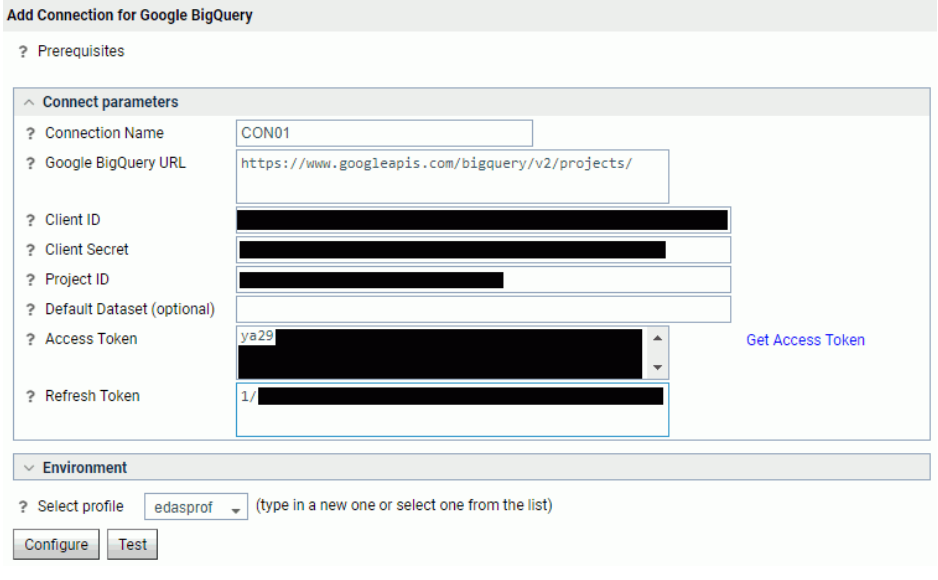
8. Enter the Google Sign In credentials and then click *Sign in*.

The Permissions Consent page opens, as shown in the following image.



9. Click *Allow*.

   You are returned to the Add Connection for Google BigQuery pane, where the Access Token field and Refresh Token field are now populated, as shown in the following image.



Note: If the Access and Refresh tokens are not populated, sign-off from your Google account and clear the browser cookies.

10. Click *Configure*.

    The Google BigQuery Adapter is added to the configured Adapters list in the navigation pane.

Reference: Connection Attributes for Google BigQuery

The following list describes the connection attributes for the Google BigQuery Adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Google BigQuery URL

The URL of the Google BigQuery API request. The default value is:

```
https://www.googleapis.com/bigquery/v2/projects/
```

Client ID

The value that identifies your application to Google BigQuery.

Obtain this value using the following steps:

1. Go to:

   ```
   https://cloud.google.com/console/project
   ```

2. Click on the Project Name for the Google BigQuery Adapter application that was previously created.
3. Click *APIs & auth* in the left pane.
4. Click *Credentials* in the left pane.
5. Use Client ID in the Client ID for web application section.

Client Secret

The value which identifies your application to Google BigQuery. This value is used in conjunction with Client ID.

Obtain this value using the following steps:

1. Go to:

   ```
   https://cloud.google.com/console/project
   ```

2. Click on the Project Name for the Google BigQuery Adapter application that was previously created.
3. Click *APIs & auth* in the left pane.
4. Click *Credentials* in the left pane.
5. Click on the OAuth 2.0 client IDs Name eg. Web client 1.
6. Use Client Secret in the Client Secret for web application section.

Project ID

The ID that is associated with the Google application.

Obtain this value using the following steps:

1. Go to:

   `https://cloud.google.com/console/project`

2. Click on the Project Name for the Google BigQuery Adapter application that was previously created.
3. Click *Home* in the left pane.
4. Use ID for the selected application.

Default Dataset

Datasets allow you to organize and control access to your tables. Specify the default dataset you want to use for loading data into Google BigQuery.

Access Token

The value that identifies the user your application is acting on behalf. Click the *Get Access Token* link to obtain this token and the Refresh Token.

In order for the Get Access Token to complete successfully, the host name used to access the WebFOCUS Reporting Server Web Console must match the host name specified for the Redirect URI in the Google BigQuery application.

A Google sign-on screen opens if you are not already logged into a Google account.

A Consent screen opens. Click *Allow*.

If an issue arises when obtaining the Access and Refresh Tokens, clear your browser cache, including cookies.

Refresh Token

The Access Token has a very short lifespan. The Refresh Token is used to obtain a new Access Token during run time.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Creating Metadata for the Google BigQuery Adapter

Create Synonym for the Google BigQuery Adapter creates the metadata used for WebFOCUS reporting.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Google BigQuery

The following liast described the synonym creation parameters for which you can supply values.

Object Type

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

`Owner/Schema`
Select an owner/schema from the drop-down list, or click Search to search.

Project

Indicates the project associated with the Google BigQuery data.

Application

Select an application directory. The default value is baseapp.

`Customize data type mappings`
If checked, opens additional fields for decomposing date fields, adding geographic roles, and selecting the data type for numeric fields and for alphanumeric fields.

`Action`
Select *Create Base Synonyms*, *Create Cluster Synonym With BV*, or *Update Base Synonyms*.

Prefix

Assign a prefix to the synonym names.

Suffix

Assign a suffix to the synonym names.

Database

Select objects for which to create synonyms. If you selected *Create Cluster Synonym With BV* as the action, the list will have facts and dimensions. If you select a fact or dimension, you can right-click it to show or add related dimensions to the list of checked items for the cluster.

Click *Create Base Synonyms*, *Create Cluster Synonym With BV*, or *Update Base Synonyms* on the ribbon.

Chapter 31

Using the Adapter for Greenplum

The Adapter for Greenplum® provides greater flexibility for SQL optimization than generic JDBC.

In this chapter:

Configuring the Adapter for Greenplum

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to a Greenplum data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one Greenplum data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Greenplum

The Adapter for Greenplum is under the SQL group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Greenplum data source. The basic syntax is:

```
jdbc:pivotal:greenplum://hostname:5432;DatabaseName=database
```

where:

hostname

Is the computer name or IP address on which the Greenplum database is located.

database

Is the name of the database.

Driver name

Name for the Greenplum driver: `com.pivotal.jdbc.GreenplumDriver`

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLGPDB SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLGPDB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the Greenplum data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the Greenplum Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or Data Management Console will encrypt the password before adding it to the server profile.

```
ENGINE SQLGPDB SET CONNECTION_ATTRIBUTES CON1
  'jdbc:xxxxxxx://hostname:port/datasource
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLGPDB SET DEFAULT_CONNECTION connection
```

where:

SQLGPDB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLGPDB SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLGPDB SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLGPDB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Greenplum Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLGPDB to identify the Adapter for Greenplum.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLGPDB [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLGPDB`

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Greenplum

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Greenplum Data Type Support* on page 831.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Greenplum synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLGPDB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Greenplum table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Greenplum Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Data Type Mapping for A256V

The Adapter for Greenplum supports data type mapping using the following setting:

```
ENGINE SQLGPDB SET CONVERSION LONGCHAR ALPHA n
```

The setting affects the mapping of all native fields that have a data type of [VAR]CHAR(32767) or TEXT. The value of *n* ranges from 1 to 32767. The default value is 256. The TEXT data type (no *n* length) can be set instead of ALPHA when necessary.

Chapter 32

Using the Adapter for HP Vertica

The Adapter for HP Vertica allows applications to access specific HP Vertica data sources. The adapter converts application requests into HP Vertica calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the HP Vertica Environment

In order to use the Adapter for HP Vertica, you must install the HP Vertica driver for whichever data source you would like to access, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the HP Vertica Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some HP Vertica Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and checking that the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server EDAPRINT log. To resolve the problem, review the HP Vertica listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for HP Vertica

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to a HP Vertica data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one HP Vertica data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for HP Vertica

The *HP Vertica* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the HP Vertica data source.

Driver name

Name for the HP Vertica driver:

```
com.vertica.jdbc.Driver
```

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLVRT SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

SQLVRT

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name.

URL

Is the URL to the location of the HP Vertica data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the HP Vertica Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLVRT SET CONNECTION_ATTRIBUTES CON1
  'jdbc:vertica://host:port'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLVRT SET DEFAULT_CONNECTION connection
```

where:

SQLVRT

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLVRT SET DEFAULT_CONNECTION SAMPLE
```

Managing HP Vertica Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLVRT to identify the Adapter for HP Vertica.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLVRT [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLVRT

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for HP Vertica

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 845.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for HP Vertica synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLVRT ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the HP Vertica table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support Report

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96

Customizing the HP Vertica Environment

The Adapter for HP Vertica provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to HP Vertica.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLVRT SET TIMEOUT {nn|0}
```

where:

`SQLVRT`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native HP Vertica driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLVRT SET PASSRECS {ON|OFF}
```

where:

`SQLVRT`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

HP Vertica Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 33

Using the Adapter for Hyperledger Fabric

The Adapter for Hyperledger Fabric allows you to seamlessly communicate with Hyperledger Fabric, which provides support for private permissioned blockchain applications. The adapter is designed for use with iWay Service Manager 8 (iSM 8), which enables users to integrate with a blockchain and utilize the intelligence of WebFOCUS, including the powerful BI and analytics platform, custom portals and dashboards, and trend analysis and reports.

Blockchain, or Distributed Ledger Technology (DLT), is a distributed record of transactions validated and stored at multiple peer locations in a network. The records form a consensus of replicated, shared, and synchronized information. The records stored in the blockchain are immutable, providing an extremely high level of security and integrity.

For more information, see the *iWay Service Manager and Blockchain Solutions Development Guide*.

In this chapter:

Preparing the Hyperledger Fabric Environment

The Adapter for Hyperledger Fabric requires the following to be configured.

- JDK 1.8 or above.
- The Hyperledger Fabric Java SDK. For information, see *Hyperledger Fabric Java SDK*. The SDK depends on a few third party libraries that must be included in your classpath. For information, see *Hyperledger Fabric Java SDK Documentation*.

Configuring the Adapter for Hyperledger Fabric

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for the Adapter for Hyperledger Fabric

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection Name
Is the logical name used to identify this set of connection attributes. The default is CON01.

PEERDEF
Is the endorser endpoint (the URL of the endorsing peer, in gRPC format).

The function of an endorsing peer is to endorse a transaction before it is committed for a particular chaincode. Every chaincode can specify an endorsement policy that may refer to a set of endorsing peers. The policy defines the necessary and sufficient conditions for a valid transaction endorsement. For example:

```
PeerAdmin@grpc://lnxx64hl2.ibi.com:7051
```

ORDERERDEF

Is the orderer endpoint (the URL of the ordering service node, in gRPC format).

The event hub is responsible for asynchronous processing of blockchain events. For example:

```
peer0@grpc://lnxx64hl2.ibi.com:7053
```

EVENTHUBDEF

Is the event hub peer endpoint (URL of peers that form the event hub, in gRPC format).

The orderers form the ordering service, a communication fabric that provides delivery guarantees. For example:

```
orderer@grpc://lnxx64hl2.ibi.com:7050
```

CHANNEL_ID

Is the channel name.

A channel is a private blockchain overlay that allows for data isolation and confidentiality. A channel-specific ledger is shared across the peers in the channel, and transacting parties must be properly authenticated to a channel in order to interact with it. For example:

```
mychannel
```

CHAINCODE_ID

Is the chaincode identifier.

A chaincode is software, running on a ledger, to encode assets and the transaction instructions (business logic) for modifying the assets. For example:

```
fabcar
```

PEER_ID

Is the member (user) name.

A member is a legally separate entity that owns a unique root certificate for the network. Network components such as peer nodes and application clients will be linked to a member. For example:

```
PeerAdmin
```

ORG_ID

Is the Membership Service Provider (MSP) Identifier.

The Membership Service Provider (MSP) refers to an abstract component of the system that provides credentials to clients and peers for them to participate in a Hyperledger Fabric network. Clients use these credentials to authenticate their transactions, and peers use these credentials to authenticate transaction processing results (endorsements). For example:

```
Org1MSP
```

CERT_LOC

Is the path to the enrollment certificate (ECert).

The ECert must have been signed by one of the Certificate Authorities (CAs) the blockchain network has been configured to trust. An enrolled user (having a signing key and ECert) can conduct chaincode instantiations, transactions, and queries with the channel. For example:

```
/qas/fabric_sdk100_sample/cert_car.pem
```

PKEY_LOC

Is the path to the private key.

```
/qas/fabric_sdk100_sample/key_car.pem
```

IBI_CLASSPATH

Are additional Java Class directories or full path names that will be available for Java Services. For example:

```
C:\directory\filename1.jar
C:\directory\filename2.jar
```

Managing Hyperledger Fabric Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Hyperledger Fabric data types.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Hyperledger Fabric

The following list describes the synonym creation parameters for which you can supply values. To create the synonym, click *Next*.

Application
Enter an application name in which to create the synonym, or click the ellipsis (...) to browse to an application folder.

Synonym Name
Enter a name for the synonym.

Function Name

Is the name of a chaincode function. For example:

```
TradingNetwork
```

Function Arguments

Are the function arguments. If there are multiple arguments, separate them with vertical bar characters (|). For example:

```
FromParty|ToParty|Amount|MessageReference
```

Chapter 34

Using the Adapter for Hyperstage

The Adapter for Hyperstage allows applications to access Hyperstage data sources. The adapter converts application requests into native Hyperstage statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Hyperstage Environment

The Adapter for Hyperstage requires the MySQL Connector/J JDBC driver. The Adapter for Hyperstage PG uses the Postgres Connector/J JDBC driver. Where required to differentiate between settings for each version, the Posgres version is referred to as the Adapter for Hyperstage PG.

Procedure: How to Prepare the Hyperstage Environment on Windows

1. Specify the location of the Postgres or MySQL Connector/J JDBC driver file in the CLASSPATH environment variable. You must specify the full location and name of the jar file.

   For example, for the mySQL version, if the jar file is located in C:\Program Files \Hyperstage\JDBC Driver, then specify:

   ```
   CLASSPATH= C:\Program Files\Hyperstage\JDBC Driver\mysql-connector-
   java-5.1.18-bin.jar
   ```

   For example, for the Hyperstage PG version, if the jar file is located in C:\ibi \srv99\homeHyperstage\hs\java, then specify:

   ```
   CLASSPATH= C:\ibi\srv99\homeHyperstage\hs\java
   \postgresql-9.2-1003.jdbc3.jar
   ```

or

```
CLASSPATH= C:\ibi\srv99\homeHyperstage\hs\java
\postgresql-9.2-1003.jdbc4.jar
```

2. Specify the location of the Java run time environment or development kit in the JAVA_HOME or JDK_HOME environment variable. Either is acceptable.

   You must specify the location where the run time environment is installed. You should see a sub-directory bin in that directory. For example, if you have the run time environment in C:\Program Files\java\jre6 then specify:

   ```
   JAVA_HOME=C:\Program Files\java\jre6
   ```

   Alternately, if you have the full development kit installed in C:\Program Files\java \jdk1.6.0_17, then specify:

   ```
   JDK_HOME= C:\Program Files\java\jdk1.6.0_17
   ```

3. Start (or restart) the server.

Note: You also need to restart the server if the driver has changed.

Procedure: How to Prepare the Hyperstage Environment on UNIX

Specify the location of several files:

1. Specify the location of the MySQL Connector/J JDBC driver files in the $CLASSPATH environment variable.

   For example, if the file is located in /usr/driver_files, you would issue the following statements for the mySQL version:

   ```
   CLASSPATH=/usr/driver_files/mysql-connector-java-5.1.18-bin.jar;
   $CLASSPATH
   export CLASSPATH
   ```

   For example, if the file is located in /usr/driver_files, you would issue the following statements for the Hyperstage PG version:

   ```
   CLASSPATH=/usr/driver_files/postgresql-9.2-1003.jdbc4.jar;$CLASSPATH
   export CLASSPATH
   ```

   To ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

   Alternatively, you could use the Web Console to specify the location:

   a. Select the *Workspace* menu option.

   b. Expand the *Java Services* folder, right-click *DEFAULT*, and select *Properties*.

The Java Services Configuration pane opens.

c. Select the *Class path* section.

d. Specify the full path to the MySQL Connector/J jar file in the IBI_CLASSPATH field.

e. Click *Save and Restart Java Services*.

2. Specify the location of the Java Development Kit's installation directory in the $JDK_HOME environment variable.

   For example, if you want to set the location of the Java Development Kit to /usr/java, you would issue the following statements:

   ```
   JDK_HOME=/usr/java
   export JDK_HOME
   ```

3. Specify the location of the Java Virtual Machine's installation directory in the $LD_LIBRARY_PATH environment variable.

   For example, if you want to set the location of the JVM to /usr/j2sdk1.4.2_01/jre/lib/i386/server, you would issue the following statements:

   ```
   LD_LIBRARY_PATH=/usr/j2sdk1.4.2_01/jre/lib/i386/server
   export LD_LIBRARY_PATH
   ```

   Note that if the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

4. Start (or restart) the server.

Note: You also need to restart the server if the driver has changed.

Hyperstage and Unicode

The Adapter for Hyperstage is implemented using JDBC. This implementation supports Unicode data stored in character fields with the CHARACTER SET set to UTF-8.

You must set the LANG environment variable in the edastart file or in a separate shell file before you start the server. For example, for American English you would export the following variable:

```
export LANG=EN_US.UTF-8
```

For details, see *Unicode Support* in the *Server Administration* manual.

Configuring the Adapter for Hyperstage

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one Hyperstage database server and to supply authentication attributes necessary to connect to the server.

You can declare connections to more than one Hyperstage database server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Overriding the Default Connection* on page 864). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The *first* SET CONNECTION_ATTRIBUTES command sets the default Hyperstage database server to be used.
- If more than one SET CONNECTION_ATTRIBUTES command declares the same Hyperstage database server, the authentication information is taken from the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Hyperstage PG

The Hyperstage PG adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Enter the location URL for the Hyperstage data source. The basic syntax is

```
jdbc:postgresql://host:port/database
```

where:

host

Is the computer name or IP address on which the Hyperstage database is located.

port

Is the port number used to access the Hyperstage database

database

Is the name of the database.

These are two examples:

```
jdbc:postgresql://localhost:8101/qatst
```

```
jdbc:postgresql://edaaix52:8101/qatst
```

Driver name

Name of the JDBC driver: `org.postgresql.Driver`

Home Directory

The full path name to the Hyperstage installation directory. For example, C:\ibi\srv77\home\hs.

Tools Directory

The full path name to the Hyperstage Tools directory. For example, C:\ibi\srv77\home\hs\bin.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Reference: Connection Attributes for Hyperstage

The Hyperstage adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Enter the location URL for the Hyperstage data source. The basic syntax is

```
jdbc:mysql://host:port/database
```

where:

host

Is the computer name or IP address on which the Hyperstage database is located.

port

Is the port number used to access the Hyperstage database

database

Is the name of the database.

These are two examples:

```
jdbc:mysql://localhost:8101/qatst
```

```
jdbc:mysql://edaaix52:8101/qatst
```

You can reference additional Hyperstage connection properties in the URL. If you wish to do so, follow these guidelines: in the URL the first property must be preceded by the ? character, and the second and subsequent properties referenced in the URL must be preceded by the & character followed immediately by an | character, as illustrated in the following example.

Suppose that you wish to add the following connection properties:

```
sessionVariables=sql_mode=PIPES_AS_CONCAT
 zeroDateTimeBehavior=convertToNull
```

Enter the URL as follows:

```
jdbc:mysql://host/database?sessionVariables=sql_mode=P
IPES_AS_CONCAT&|
 zeroDateTimeBehavior=convertToNull
```

Note: The URL must be entered as a single line, without a space after the | character.

Driver name

Name of the JDBC driver: `com.mysql.jdbc.Driver`

Home Directory

The full path name to the Hyperstage installation directory. For example, C\ibi\Hyperstage.

Tools Directory

Hyperstage Tools directory.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

For explicit authentication:

```
ENGINE SQLHYP SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

For password passthru authentication:

```
ENGINE SQLHYP SET CONNECTION_ATTRIBUTES connection/
```

where:

SQLHYP

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

For the Postgres version of the adapter, use SQLHYPG.

connection

The name of the connection. You can give any name to the connection that you want.

userid

Is the primary authorization ID by which you are known to Hyperstage.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Hyperstage database server named TEST with an explicit user ID and password:

```
ENGINE SQLHYP SET CONNECTION_ATTRIBUTES TEST/USERA,PWDA
```

The following SET CONNECTION_ATTRIBUTES command connects to the Hyperstage database server named TEST using password passthru authentication:

```
ENGINE SQLHYP SET CONNECTION_ATTRIBUTES TEST/
```

Authenticating a User

There are two methods by which a user can be authenticated when connecting to a Hyperstage database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Database or Password Passthru.** User ID and password received from the client application are passed to the Hyperstage database server for authentication.

When a client connects to the server, the user ID and password are passed to Hyperstage for authentication and are not authenticated by the server. To implement this type of authentication, start the server with security turned off. The server allows the client connection, and then stores an encrypted form of the client connection message to be used for connection to a Hyperstage database server at anytime during the lifetime of the server agent.

Overriding the Default Connection

Once all Hyperstage connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific Hyperstage connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.
- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax: How to Select a Connection to Access

```
ENGINE SQLHYP SET DEFAULT_CONNECTION [connection]
```

where:

`SQLHYP`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

For the Postgres version of the adapter, use SQLHYPG.

`connection`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the last command will be the active connection name.

- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example: Selecting a Connection to Access

The following SET DEFAULT_CONNECTION command selects the Hyperstage database server named SAMPLESERVER as the default Hyperstage database server:

```
ENGINE SQLHYP SET DEFAULT_CONNECTION SAMPLESERVER
```

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for the datasource_name.

Controlling Connection Scope

This topic explains how to set the scope of logical units of work using adapters. This is accomplished by the SET AUTODISCONNECT command.

A connection occurs at the first interaction with the declared database server.

Syntax: How to Control the Connection Scope

```
ENGINE SQLHYP SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLHYP

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Hyperstage Metadata

This topic describes how to use CREATE SYNONYM for Hyperstage data sources. It also describes Hyperstage data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each Hyperstage table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Hyperstage

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Object name

Selecting this option adds the Object Name parameters to the screen.

Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Hyperstage Data Type Support* on page 872.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Type

The object type (Table, View, and so on).

Table name

Is the name of the underlying object.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Hyperstage synonym comprises a Master File and an Access File. This is a synonym, using the mySQL version of the adapter for the table nf29004.

Generated Master File nf29004.mas

```
FILENAME=NF29004, SUFFIX=SQLHYP  , $
SEGMENT=NF29004, SEGTYPE=S0, $
FIELDNAME=DIVISION4, ALIAS=DIVISION4, USAGE=I11, ACTUAL=I4, $
FIELDNAME=DIVISION_NA4, ALIAS=DIVISION_NA4, USAGE=A25, ACTUAL=A25,
MISSING=ON, $
FIELDNAME=DIVISION_HE4, ALIAS=DIVISION_HE4, USAGE=I11, ACTUAL=I4,
MISSING=ON, $
```

Note: For the Postgres version of the adapter, the SUFFIX value would be SQLHYPG.

Generated Access File nf29004.acx

```
SEGNAME=NF29004, TABLENAME=NF29004, CONNECTION=CON1, KEYS=0, $
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Hyperstage tablename. The tablename may include owner (schema) name. For example, TABLENAME=[*owner.*]*tablename*

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. Since Hyperstage does not support indexes, the default value is 0. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❏ KEYFLD is the FIELDNAME of the common column from the parent table. ❏ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Keyword	Description
AUTO INCREMENT	Not applicable for Hyperstage.
START	Not applicable for Hyperstage.
INCREMENT	Not applicable for Hyperstage.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Not applicable for Hyperstage.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Hyperstage Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96

Customizing the Adapter for the Hyperstage Environment

This topic describes how to set the adapter for the Hyperstage environment.

PASSRECS

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Set PASSRECS

```
ENGINE SQLHYP SET PASSRECS {ON|OFF}
```

where:

`SQLHYP`

Indicates adapter. You can omit this value if you previously issued the SET SQLENGINE command.

For the Postgres version of the adapter, use SQLHYPG.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE SQLHYP SET ISOLATION {RU|RC|RR|SE}
```

where:

SQLHYP

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

For the Postgres version of the adapter, use SQLHYPG.

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Hyperstage Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Chapter 35

Using the Adapter for i Access

The Adapter for i Access allows applications to access system i data sources using odbc and jdbc from windows and unix environments. The adapter converts application requests into native ODBC/JDBC calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the i Access Environment (ODBC)

In order to use the Adapter for i Access, you must install the IBM i Access for Windows client on the box where the server will be running.

Preparing the i Access Environment (JDBC)

In order to use the Adapter for JDBC, you must install the JDBC driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the JDBC Environment on Windows and UNIX

1. Identify the location of the JDBC Driver files by adding them to the environment variable *CLASSPATH* before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   On UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

Similarly, on Windows, you would issue the command:

```
set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
```

Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

2. Start, or restart. the server.

   Note that you also need to restart the server if the driver has changed.

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace and checking that the special services section displays *JSCOM3 active*.

   ```
   edastart -show
   ```

   If the JSCOM3 service is not active, a *fail to start* message will typically also be inserted in the server EDAPRINT log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for i Access

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to the i database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one i Access database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the i Access Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for i Access (ODBC)

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

SYSTEM

Name of the machine where the IBM i system is running.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Default Library (optional)

Specifies the IBM i libraries to add to the server job library list. The library list is used for resolving unqualified stored procedure calls and finding libraries in catalog API calls.

Additional connection string keywords (optional)

Refer to IBM i access documentation for all available connection string keywords used to change the behavior of the ODBC connection.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually (ODBC)

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to i Access, at connection time, for authentication.

For User/System DSN:

```
ENGINE SQLIIA SET CONNECTION_ATTRIBUTES connection DSN_SAMPLE/
userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to IBM i, at connection time, for authentication.

```
ENGINE SQLIIA SET CONNECTION_ATTRIBUTES connection DSN_SAMPLE/
```

Trusted authentication. The adapter connects to IBM i as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE SQLIIA SET CONNECTION_ATTRIBUTES connection DSN_SAMPLE/,
```

Example: Declaring Connection Attributes (ODBC)

For User/System DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the i Access DSN server named SAMPLE with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify OS4V71P1 connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLIIA SET CONNECTION_ATTRIBUTES CON1 SAMPLE/MYUSER,PASS
```

Reference: Connection Attributes for i Access (JDBC)

The IBM i Access adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the JDBC data source.

Driver name

Name for the JDBC driver.

For information, see the driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. This value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Syntax: How to Declare Connection Attributes Manually (JDBC)

```
ENGINE SQLIIA SET CONNECTION_ATTRIBUTES connection 'url'/userid,password
```

where:

`SQLIIA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`url`

Is the URL to the location of the JDBC data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes (JDBC)

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLIIA  SET CONNECTION_ATTRIBUTES CON1
'jdbc:xxxxxxx://hostname:port/datasource'/MYUSER,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

  ```
  ENGINE SQLIIA SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

```
CONNECTION=DSN_A
```

If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

```
ENGINE SQLIIA SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLIIA SET DEFAULT_CONNECTION connection
```

where:

`SQLIIA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: **Selecting the Default Connection**

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLIIA SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: **How to Control the Connection Scope**

```
ENGINE SQLIIA SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLIIA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing i Access Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLIIA to identify the Adapter for i Access.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLIIA [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLIIA

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.

- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for i Access

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *i Access Data Type Support* on page 891.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for i Access synonym comprises a Master File and an Access File. This is a synonym for the table TYEAR.

Master File

```
FILENAME=TYEAR, SUFFIX=SQLIIA  , $
  SEGMENT=TYEAR, SEGTYPE=S0, $
    FIELDNAME=MONTH, ALIAS=MONTH, USAGE=A6V, ACTUAL=A6V, $
    FIELDNAME=YEAR, ALIAS=YEAR, USAGE=A6V, ACTUAL=A6V,
      MISSING=ON, $
    FIELDNAME=QUARTER, ALIAS=QUARTER, USAGE=A6V, ACTUAL=A6V,
      MISSING=ON, $
    FIELDNAME=MONTH_CAPTION, ALIAS=MONTH_CAPTION, USAGE=A6V, ACTUAL=A6V,
      MISSING=ON, $
```

Access File

```
SEGNAME=TYEAR,
   TABLENAME=DB999.TYEAR,
   CONNECTION=CON01,
   KEY=MONTH, $
  INDEX_NAME=TYEARIX,
      INDEX_UNIQUE=Y,
      INDEX_COLUMN=MONTH,
      INDEX_ORDER=ASC, $
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the table name. For IBM i, the synatx is: TABLENAME=[*library*.]*tablename*

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection*
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

i Access Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the i Access Environment

The Adapter for i Access provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to IBM i.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLIIA SET TIMEOUT {nn|0}
```

where:

`SQLIIA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

0

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLIIA SET PASSRECS {ON|OFF}
```

where:

SQLIIA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE SQLIIA SET ISOLATION {RU|RC|RR|SE}
```

where:

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

i Access Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 36

Using the Adapter for CA-IDMS/DB

The Adapter for CA-IDMS/DB for z/OS allows applications to access IDMS/DB data sources. The adapter converts application requests into native IDMS/DB statements and returns optimized answer sets to the requesting application.

Note: In the remainder of this topic, the following terms all refer to the Adapter for CA-IDMS/DB:

- CA-IDMS/DB
- IDMS/DB
- Adapter for IDMS/DB

In this chapter:

Preparing the IDMS/DB Environment

Prior to configuring the Adapter for IDMS/DB using the Web Console, it is necessary to edit the ISTART JCL that is used to initiate the server.

Procedure: How to Edit the ISTART JCL

1. Allocate the IDMS.LOADLIB and IDMS.DBA.LOADLIB libraries to the server STEPLIB DDNAME allocation.
2. Allocate DDNAME SYSCTL to the IDMS.SYSCTL data set.
3. Allocate DDNAME SYSIDMS to the IDMS.SYSIDMS data set.

Configuring the Adapter for IDMS/DB

You can configure the Adapter for IDMS/DB from the Web Console or the Data Management Console.

The integrity of the IDMS/DB data source is not jeopardized because the adapter provides read-only access. Access to the data source is restricted by the security package in use on the z/OS system (such as RACF) and by the CA-IDMS/DB security features.

Both CA-IDMS/DB and the server provide security mechanisms to ensure that users have access to only those objects for which they have authorization.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

IDMS/DB Overview and Mapping Considerations

CA-IDMS/DB databases, both network and LRF-based, correspond to hierarchical and relational data sources. The concepts discussed in this section affect the way IDMS/DB files are described to the server.

Mapping Concepts

IDMS/DB is a network database management system that is accessed by a subschema (the equivalent in the server is a Master File). IDMS/DB provides two methods of retrieving records within a subschema from an application program:

- Network access (DML)
- LRF-based access (LRF)

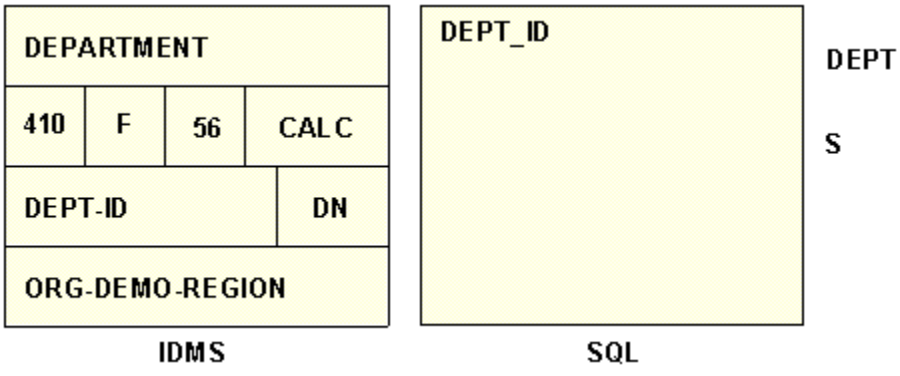
Data Management Language (DML) access is the traditional method of IDMS/DB database navigation. It is the network navigation facility. Each physical record is retrieved separately. An application program enters the IDMS/DB database at a particular IDMS/DB record type (the equivalent in the server is a segment) and searches the set connections to retrieve the required data. For mapping concepts that apply to network record types, see *Summary of Network Relationships* on page 909.

Logical Record Facility (LRF) access, available as of IDMS/DB Release 5.7, is the relational-like method of access. LRF provides software to dynamically create flat records from one or more network record types at run time. For mapping concepts that apply to LRF-based records, see *Logical Record Facility Concepts* on page 907.

Network Concepts

An IDMS/DB record type is described to the server as a segment. Since DML retrieval is record-oriented rather than field-oriented, the Master File must list the fields in the same order as they appear on the IDMS/DB record type. However, a Master File does not have to list every field of a particular record type; you may omit fields after a given field. For example, your description may list the first four fields of a 10-field IDMS/DB record type. You can use IDMS/DB field names in your Master File up to a maximum of 48 characters.

The following diagram illustrates how a record type is described as a segment. The record type DEPARTMENT contains the field-type DEPT-ID. Its corresponding server segment DEPT contains a field named DEPT_ID.



Repeating fields on an IDMS/DB record type are defined as OCCURS segments.

Network record types may be related to each other. These relationships may be physical (using set connections) or logical (achieved through an index or CALC field). The Adapter for IDMS/DB supports both physical and logical relationships.

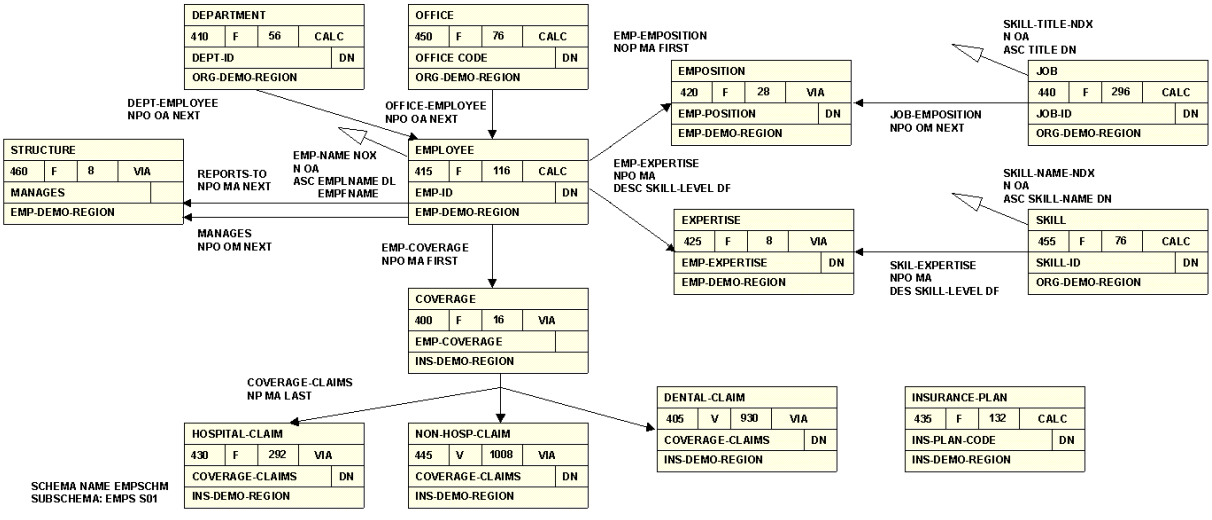
Set-Based Relationships

In an IDMS/DB database, physical relationships between record types are achieved with pointers that correspond to IDMS/DB sets. A set implements a one-to-many relationship between record types. The server equivalent of a set is the parent/descendant relationship between segments. In an IDMS/DB set, one record type acts as the owner (the one side of the relationship) and one or more record types act as the members (the many side of the relationship). A single IDMS/DB record type can participate in several set relationships as either the owner or the member.

The IDMS/DB representation of record types and set relationships within a database is called a Bachman diagram, also known as a data structure diagram. In a Bachman diagram, a record type is depicted as a box; a set, as a line with an arrow. Set names appear as labels beside the arrows. The box that the arrow points to is the member record type. Triangles indicate indexes. The next diagram is the Bachman diagram for the IDMS/DB network subschema EMPSS01. Sections of this diagram are referenced throughout these sections.

The following kinds of set-based relationships are depicted: simple set, common owner, common member, multi-member, bill-of-materials (simple and multi-tiered), and loop structures.

All relationships correspond to server structures and are explained following the diagram:



Simple Set

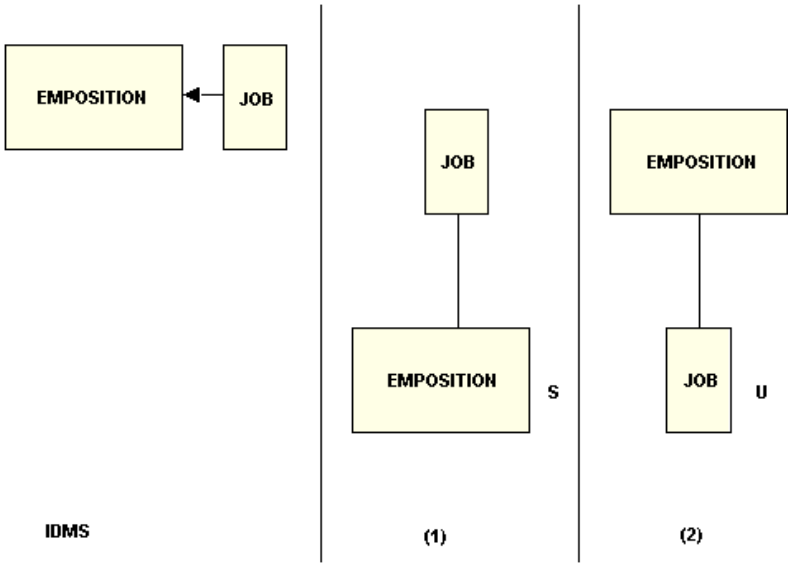
The following diagram illustrates the basic mapping principle of a simple set: an IDMS/DB record type corresponds to a segment; a set relationship corresponds to a parent/descendant relationship.

This figure also illustrates a second principle: an IDMS/DB structure can have more than one representation as a hierarchy. The type of parent/descendant relationship required in the Master File depends on whether the owner or the member record type is designated as the parent segment.

The JOB-EMPOSITION set has two possible representations:

1. It shows the JOB record type, the owner in the set, mapped to the server as the root segment. Since a member record type is multiply-occurring (for example, several instances of EMPOSITION records per JOB instance, indicating that many positions share one job title and description), the EMPOSITION record type is displayed as a non-unique descendant.

2. It depicts the reverse. The EMPOSITION record type is the root segment and JOB is the unique descendant, since an EMPOSITION instance can have only one owner (only one job title and description per position).



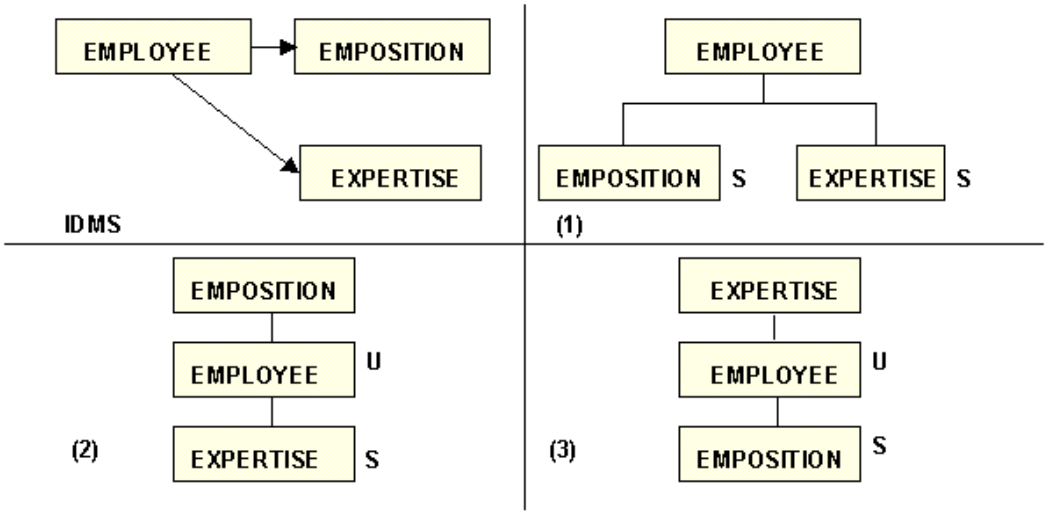
Common Owner

Given the rules in the previous example, consider a more complex scenario called a common owner. A common owner structure contains a record type that is the owner of two or more record types. Several representations are possible.

For example, the diagram below depicts three ways to describe the EMPLOYEE, EXPERTISE, and EMPOSITION structure in one Master File:

1. It shows the EMPLOYEE record type, the owner in both sets, mapped to the server as the root segment of EMPOSITION and EXPERTISE. Since an EMPLOYEE record can have many EMPOSITION and EXPERTISE records, both descendant records are non-unique.
2. It depicts the EMPOSITION record type as the root segment of EMPLOYEE, which acts as the parent of EXPERTISE. Since an EMPOSITION record can have only one owner, EMPLOYEE is a unique descendant; EXPERTISE is a non-unique descendant of EMPLOYEE.

3. It depicts the EXPERTISE record type as the parent of EMPLOYEE, which acts as the parent of EMPOSITION. EMPLOYEE is a unique descendant and EMPOSITION is a non-unique descendant of EMPLOYEE.



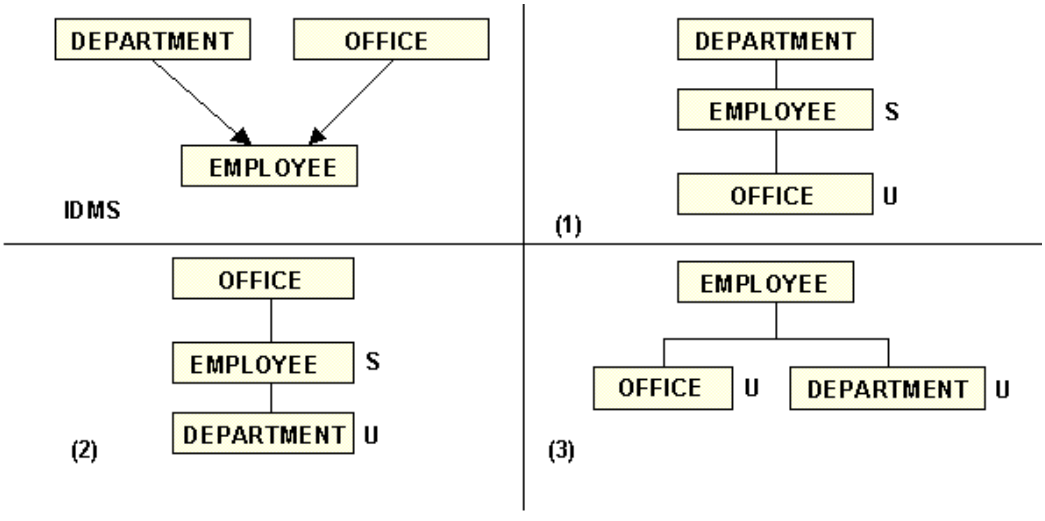
Common Member

When an IDMS/DB record type is a member of two or more sets, the association of the owner record type as the parent segment must be abandoned for one or more sets, because a segment can have only one parent. The diagram below displays the possible interpretations for this IDMS/DB configuration.

In the diagram, the EMPLOYEE record type is a common member in the DEPT-EMPLOYEE and the OFFICE-EMPLOYEE sets. This structure can be described to the server in three ways:

1. It shows DEPARTMENT as the root segment with EMPLOYEE as its non-unique descendant. OFFICE is the unique descendant of EMPLOYEE.
2. It depicts the reverse: OFFICE is the root segment; EMPLOYEE is its non-unique descendant; and DEPARTMENT is the unique descendant of EMPLOYEE.

3. It shows the only other alternative: EMPLOYEE is the parent of OFFICE and DEPARTMENT. Both are unique, since an EMPLOYEE can belong to only one OFFICE and DEPARTMENT.



Notice that the rules for simple sets are still valid:

- If the owner record type is the parent segment, the member record type as a descendant segment is non-unique.
- If the member record type is the parent segment, the owner record type as a descendant segment is unique.
- A member or an owner record type may act as a root segment (the server ignores whether the root segment is unique or non-unique).

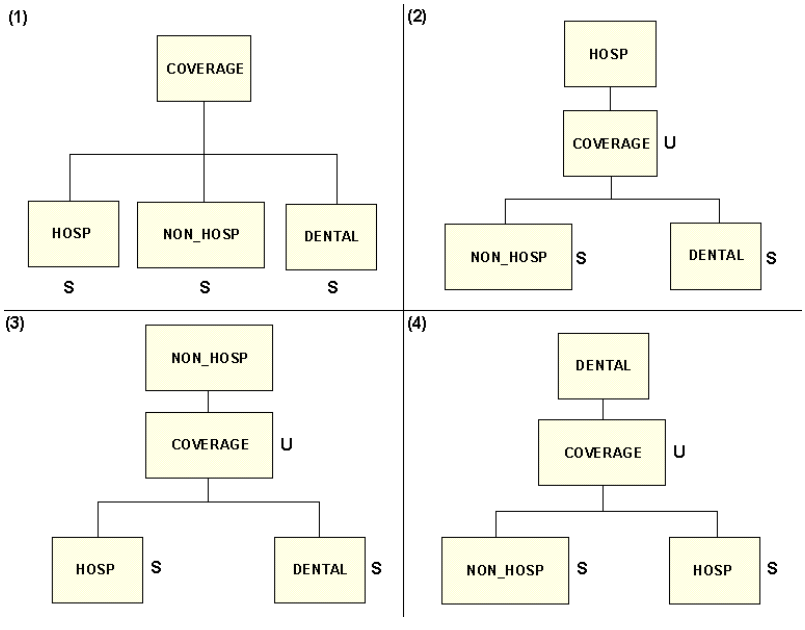
It may be helpful to think of the hierarchical depiction of a network structure as its navigational path. From an IDMS/DB standpoint, panel 1 of the previous diagram shows that the IDMS/DB database can be entered at the DEPARTMENT record type. The corresponding EMPLOYEE record occurrences for a DEPARTMENT record occurrence can be obtained by searching the DEPT-EMPLOYEE set. For each EMPLOYEE record occurrence, obtaining the owner in the OFFICE-EMPLOYEE set retrieves the corresponding OFFICE record occurrence. This is a three-segment retrieval hierarchy that maps to server descriptions.

Multi-Member

When there is more than one member record type, the set is called a multi-member set. A multi-member set is represented in the Master File exactly like a common owner set. The fact that the two relationships are based on the same set is stated in the Access File.

For example, to describe the COVERAGE record type and its three members of the COVERAGE-CLAIMS set, you may choose one of four ways as depicted in the following diagram:

1. The first panel shows the COVERAGE record type, owner of the multi-member set, as the root segment. Since several instances of CLAIMS can be reported against one insurance policy (COVERAGE), each member is a non-unique descendant.
2. The second panel depicts HOSPITAL-CLAIM as the parent of COVERAGE, and the other two member record types as descendants of COVERAGE. In each case, a claim can be reported against only one insurance policy. This explanation applies to panels 3 and 4 as well.



Bill-of-Materials

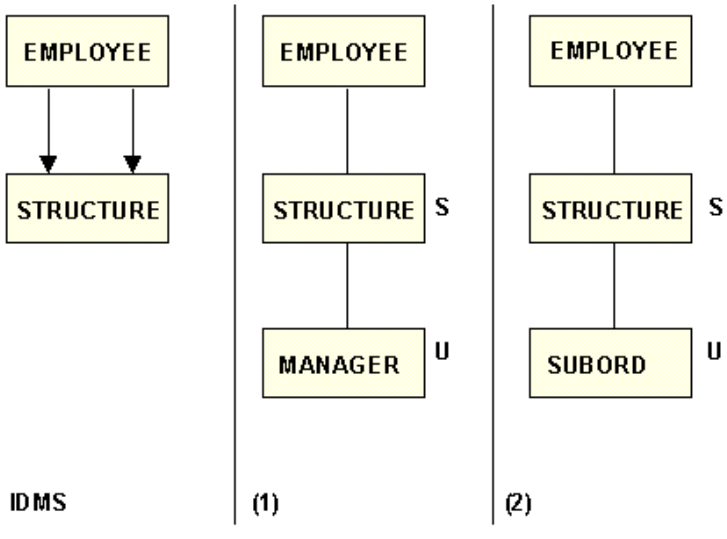
Bill-of-materials structures are classified as simple or multi-tiered. In this section, the simple version is discussed first.

An instance of two record types being linked by more than one set is called a bill-of-materials structure. This structure describes a many-to-many relationship between record occurrences of the same record type. The member record type is the junction record type between the two related owners.

For a simple bill-of-materials structure, the server requires that the owner record type be represented as two or more segments with different field names for the identical fields. This ensures that, at retrieval time, the field names specified in the user's request will provide the proper navigational path.

In the following diagram, the EMPLOYEE and STRUCTURE record types are connected by two sets. This simple bill-of-materials structure can be described two ways:

1. As an employee-to-manager relationship. The EMPLOYEE record type is the parent segment of the non-unique STRUCTURE segment using the REPORTS-TO set. The STRUCTURE record type, in turn, is the parent segment of the unique MANAGER segment using the MANAGES set (the MANAGER segment duplicates the EMPLOYEE segment and its fields are renamed).
2. As an employee-to-subordinate relationship. The EMPLOYEE record type is the parent segment of the non-unique STRUCTURE segment using the MANAGES set. The STRUCTURE record type is the parent segment of the unique SUBORD segment using the REPORTS-TO set (the SUBORD segment duplicates the EMPLOYEE segment and its fields are renamed).



The previous diagram represents a two-tiered employee-to-employee relationship. Multi-tiered relationships are extended bill-of-materials structures. Multi-tiered relationships are created and used for different levels of answer sets. The number of levels or tiers should be kept to a minimum.

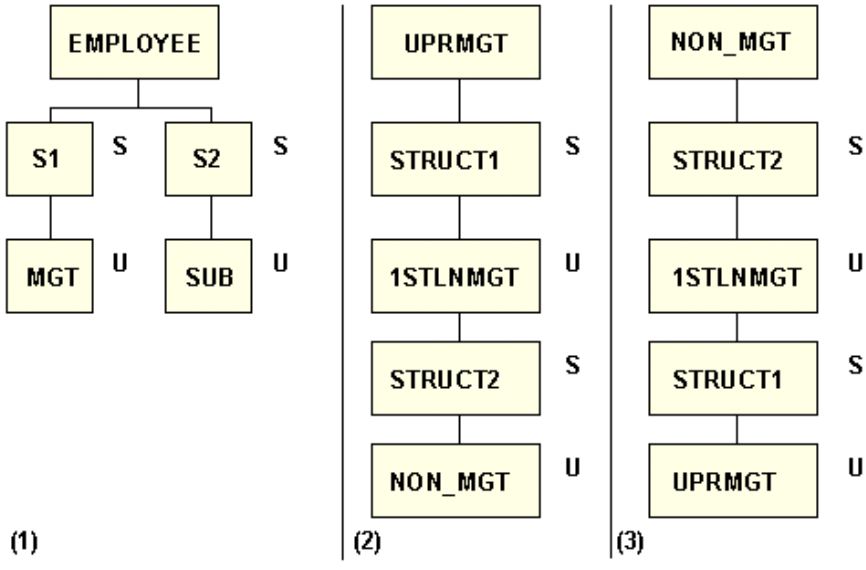
To determine the number of segments required to describe an n-tiered relationship, use this formula:

```
Number of segments = (2 x number of tiers) - 1
```

The following diagram is a three-tiered version of the previous diagram:

1. The first panel combines both views with EMPLOYEE as the parent of two non-unique descendants, S1 and S2. Both S1 and S2, in turn, are parents of a unique descendant, MGT and SUB, respectively (S1 and S2 describe junction records that point to MGT and SUB).

2. The second panel shows a three-tiered relationship between employees implemented in a five-segment single-path hierarchy. The segments UPRMGT (upper management), 1STLNMGT (first-line management), and NON_MGT (non-management) all describe the EMPLOYEE segment but have renamed fields (like S1 and S2 above, STRUCT1 and STRUCT2 contain renamed fields that point to descendant segments).
3. The third panel depicts the opposite of panel 2.

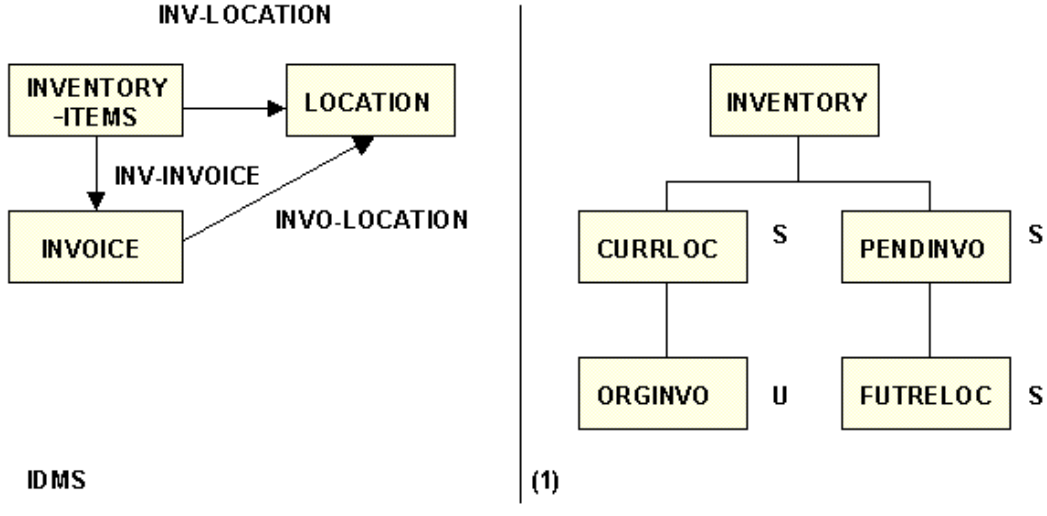


Loop Structures

Loop structures in IDMS/DB implement complicated relationships among record types. For the server depiction of a loop, you must select a record type to be the parent in the relationship.

Suppose you had a loop structure like the one below. An INVOICE record occurrence has an owner record occurrence (INVENTORY-ITEMS) only if the invoice order is pending or has not been delivered. The INVO-LOCATION set lists the location where an invoice item will be or was delivered.

In panel 1 in the following diagram, the server translates this structure as multi-tiered, with one tier of renamed segments. The INVENTORY-ITEMS record type, in this case, acts as the root INVENTORY. The LOCATION record type is mapped as the CURRLOC segment that lists the location of items currently in stock. The INVOICE record type is mapped as the PENDINVO segment that lists pending invoice orders. Segments ORGINVO and FUTRELOC are required segments that rename the data in record types INVOICE and LOCATION. The segment ORGINVO contains historical information for an inventory item. The segment FUTRELOC indicates where an ordered item will be delivered.



CALC-Based and Index-Based Relationships

Logical relationships, unlike physical ones, are based on the occurrence of the same data value in two different record types. To make the parent/descendant connection, the Adapter for IDMS/DB uses CALC fields or indices to locate the related record occurrence(s). The related fields are not required to have the same name in both record types, but the field format must be the same.

Note:

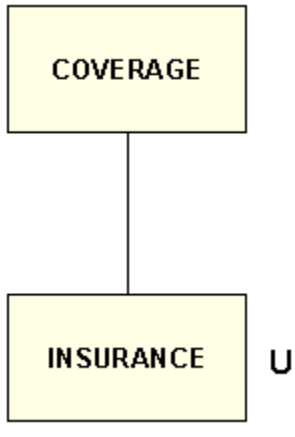
- WHEREs on CALC fields and indices are also used to generate CALC and index calls to IDMS/DB.
- CALC or index fields can also be GROUPNAMEs, consisting of fields contiguous in both parent and descendant segments. The format types and lengths of the GROUP and its fields must be comparable in both parent and descendant segments.

Like set-based descendants, CALC- and index-based descendants are unique or non-unique, but this depends largely on how the DUPLICATES parameter is specified in the Access File (the SEGTYPE parameter for the descendant must also reflect the DUPLICATES parameter; for example, CLCDUP=N, SEGTYPE=U). The server treats unique descendants in the same manner regardless of what underlies the parent/descendant relationship: set, index, or CALC field.

The COVERAGE and INSURANCE-PLAN record types in the diagram in *Set-Based Relationships* on page 898 illustrate a logical relationship. The field PLAN-CODE is common to both record types. The parent/descendant relationship can be described to the server if the:

- INSURANCE-PLAN record type has an index on PLAN-CODE.
- INSURANCE-PLAN record type is an IDMS/DB CALC record type with PLAN-CODE as the CALC key.

The diagram below depicts the server representation of the diagram in *Set-Based Relationships* on page 898 that uses the CALC field method. The server interprets COVERAGE as the parent segment and INSURANCE-PLAN as the descendant. Since the DUPLICATES parameter is set to N (CLCDUP=N), the INSURANCE segment is unique.



Logical Record Facility Concepts

The Adapter for IDMS/DB supports two kinds of LRF-based records:

- Logical Records (LR) are built from network record types. They are not physically stored as such; instead, they are defined within a subschema using DBA-supplied navigational paths. Logical Records are created at execution time from real database records-types based on DBA-supplied navigational paths and the SELECTs passed to LRF from the server.

- Automatic System Facility (ASF) records are created by end-users with the Automatic System Facility, a menu-driven front end to LRF. This Facility generates subschemas and navigational paths based on the user's menu selection.

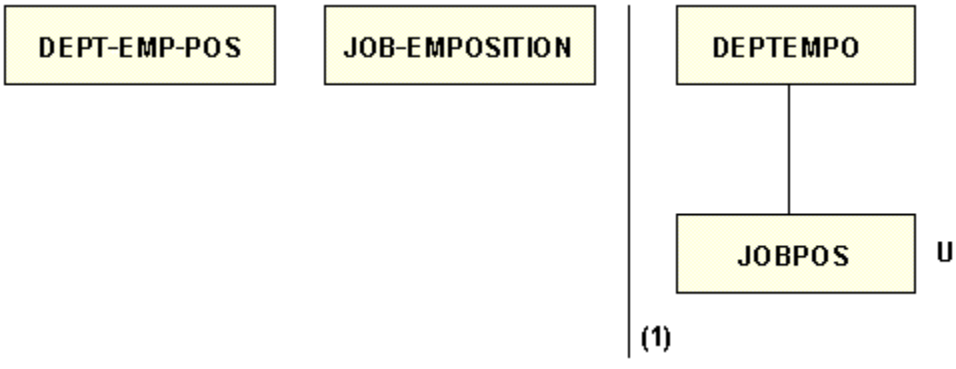
The Adapter for IDMS/DB uses the IDMS/DB Logical Record Facility (LRF) to retrieve and create ASF and LR records. In general, the LRF-based records can contain information from several sources (both DML created and ASF generated) that are located in several database areas. All fields, records, and areas, however, must be defined to the same subschema.

LRF-based records may be related to each other using an embedded cross-reference. Your Master File can list up to 64 related LRF-based records. Access Files can list an unlimited number of subschemas. However, the server accesses up to 16 subschemas per request.

LRF Records as Descendants

With the Adapter for IDMS/DB, you may create a parent/descendant relationship between two LRF records if they share a common field or GROUP. Field lengths and formats must be the same in both segments. The shared fields are specified in the descendant segment declaration of the Access File as the values of the KEYFLD/IXFLD pair.

For example, in the diagram below, the LRF record DEPT-EMP-POS is represented as the root segment DEPTEMPO; the LRF record JOB-EMPOSITION as the unique segment JOBPOS. The related fields are POS_STRT_DR2 (the IXFLD value) and POS_STRT_DT1 (the KEYFLD value).



The restriction that the shared field in the descendant record be a CALC or an indexed field does not apply to LRF records. However, this kind of embedded cross-reference has one restriction: the IDMS/DB path group for an LRF record that acts as the descendant segment must contain a SELECT clause to process the implied WHERE clauses.

Syntax: **How to Implement a Parent/Descendant Relationship With SELECT**

SELECT clauses are maintained by your DBA and should already be available in your subschema. The following SELECT clauses can successfully implement a parent/descendant relationship. They are listed in descending order of efficiency

1. `SELECT FOR FIELDNAME EQ` *`fieldname`*
2. `SELECT USING INDEX` *`indexname`*
3. `SELECT FOR FIELDNAME` *`fieldname`*
4. `SELECT FOR ELEMENT` *`elementname`*
5. `SELECT`

where:

`fieldname`

Is the joined-to IDMS/DB field name on the descendant record type.

`indexname`

Is the index set name to the joined-to field.

`elementname`

Is the physical record type on which the joined-to field is stored.

Note:

- If no specific SELECT clause exists, a null SELECT clause (item 5) must be available to process an answer set.
- LRF records that return partial record occurrences and user-defined path status should not be used. If the adapter encounters a user-defined path status, the status is interpreted as LR-ERROR and the retrieval process is halted with messages EDA967 and EDA949.
- Do not confuse the use of the word SELECT with the SQL SELECT statement. In the above example, the word SELECT is used to refer to the IDMS/DB internal meaning.

Summary of Network Relationships

As illustrated in the previous examples, there are four ways to implement a network relationship. Each can be described as a parent/descendant relationship:

- Owner/member

- Member/owner
- Field or GROUP/CALC field
- Field, GROUP/SPF, or Integrated Index

All four can be intermixed in one Master File. The actual underlying connection (set, CALC, index) between parent and descendant is not apparent to end users who specify field names. In addition, a Master File can list record types from multiple subschemas, databases, and dictionaries.

In server processing, the identity of the record type behind a segment is invisible. The retrieval technique is followed in all cases as if all segments were represented by distinct records. Within IDMS/DB, currencies are maintained by storing the DBKEYs of record types and retrieving the record occurrence again, when needed.

The top-to-bottom order in which segments are defined (the chains of parent/descendant relationships) corresponds to structural relationships among the IDMS/DB record types. This top-to-bottom order is logically significant; the left-to-right order, on the other hand, is not.

Managing IDMS/DB Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IDMS/DB data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each IDMS/DB file that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

For the Adapter for IDMS/DB to access IDMS/DB files, you must describe each IDMS/DB file you use in a Master File and Access File. The logical description of an IDMS/DB file is stored in a Master File, which describes the field layout.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for IDMS/DB

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Dictionary Name

Type a data dictionary name.

Database Name

Type a database name.

Subschema Name

From the drop-down list of subschemas, select a subschema.

Select Root Record Name and Child Records

1. Choose the first record of the selected subschema to be described and click Add. This record will be the root segment in the generated Master File.

   Note: If a field name contains dashes (-) as part of the name, they are converted to underscores (_) in the generated Master File.

   It is recommended that the root record-type have a set relationship with a CALC or index field.

2. If you want to add additional levels of records to this structure, right-click the record name and then choose Add Child from the pop-up menu. Records that have no available children display a red diamond next to the record name. Those with available children display a stack of green squares.

   A list of available child segments is displayed. Check each record that you want to be a child of the previous record. To select all of the listed records, check the Record Name check box.

3. When you have selected the child records you want to add to the structure, click Add. You can repeat this process of selecting children and creating additional levels in the structure until no children are available.

4. When you are finished selecting children, right-click the record name and choose Save from the pop-up menu.

Application

Select an application directory. The default value is baseapp.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Synonym Name

If necessary, edit the default name for the synonym that will be created.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Master Files for IDMS/DB

This section explains the rules for making an IDMS/DB data source accessible to the server using the Master File. The Access File is an extension of Master File syntax that provides information to map DML record types and LRF-based records, including record and area names and, where needed, set or field name information.

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.

The following sections summarize the syntax for each type of declaration, and then describe each attribute.

File Attributes

Master Files begin with a file declaration that names the file and describes the type of data source; in this case, an IDMS/DB data source. The file declaration has two attributes, FILENAME and SUFFIX.

Syntax: How to Identify a Master File

```
FILE[NAME]=name, SUFFIX=IDMSR [,$]
```

where:

name

Is any 1- to 8-character name. On z/OS, the Master File is the member name within the PDS allocated to DDNAME MASTER.

IDMSR

Indicates that the Adapter for IDMS/DB is required for data retrieval.

Segment Attributes

Each IDMS/DB segment described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE.

- *SEGNAME*
- *SEGTYPE*
- *PARENT*
- *CRFILE (Cross-referenced File)*
- *OCCURS and POSITION*

SEGNAME

The SEGNAME value assigned to DML record types and LRF records are suggestive names – not necessarily the IDMS/DB record names. The segment name may be a maximum of eight characters and must be unique within a given Master File. If the same IDMS/DB record type is viewed in two different contexts (for example, a loop structure), two segment declarations are required.

SEGTYPE

The SEGTYPE keyword indicates whether a segment occurs once (SEGTYPE=U) or many times (SEGTYPE=S). It is used as follows:

- For a root segment, SEGTYPE has no meaning and may be omitted entirely.
- For a descendant segment, SEGTYPE values can be S or U.
- For descendant segments with set-based relationships, SEGTYPE indicates whether the segment acts as an owner or a member. If the descendant segment is the owner record type, the SEGTYPE value is U. If the descendant is the member, the SEGTYPE value is S.
- For descendant segments with CALC-based or index-based relationships, SEGTYPE values may be S or U depending on the DUPLICATES (CLCDUP or IXDUP) value.
- For descendant segments with LRF-based relationships, SEGTYPE values may be S or U.

PARENT

The PARENT keyword is required for descendant segment declarations. The PARENT value names the descendant's parent segment. This keyword is not specified for the root declaration.

CRFILE (Cross-referenced File)

The CRFILE keyword is specified only for segments that are described remotely in another Master File. The field descriptions for these segments are, in effect, copied into the Master File at execution time. Remote descriptions are discussed in *Remote Descriptions* on page 920.

OCCURS and POSITION

The OCCURS and POSITION attributes are specified only when the segment corresponds to an intra-record structure, such as a COBOL OCCURS or OCCURS DEPENDING clause. These keywords are described in detail in *Intra-record Structures: OCCURS Segment* on page 921.

Field Attributes

Each segment consists of one or more fields. The Master File need not describe all fields from a segment. The attributes are:

- *FIELDNAME*
- *ALIAS*
- *USAGE*
- *ACTUAL*
- *GROUP Fields*
- *IDMS/DB Database Key*

To describe a field in the Master File, you must specify the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. These attributes are discussed in this section. Note that the adapter does not support the MISSING attribute.

FIELDNAME

The FIELDNAME keyword may contain any name, not necessarily an IDMS/DB field name. It can be a maximum of 48 characters. The name that you assign must be unique, because in the server, data is referenced through field names. If the same IDMS/DB record type viewed in different contexts underlies two or more segments, different field names must be specified in separate segment descriptions.

Due to the record-oriented nature of IDMS/DB, the fields are described for each segment in the same order as they appear in the subschema record area. Bytes skipped for field alignment or for fields you want to omit must be described as dummy fields of the appropriate length.

For example, a 1-byte alphanumeric field is followed by a double-precision floating point field:

```
SEGNAME=EMPSTATUS,$
  FIELD=STATUS_CODE,SCODE ,A1,A1,$
  FIELD=            ,     ,A7,A7,$
  FIELD=GROSS_SALES,GSALES,D12.2,D8,$
```

The same name can be used for all dummy fields; in the example above, blanks are used.

You do not need to describe all fields of the record type or LRF record; only an initial set, starting with the first field and continuing up to the last field of interest. For variable-length record types, treat the fixed-length portion as one segment and the variable portion as another segment, as described with the OCCURS attribute. For information about OCCURS segments, see *Intra-record Structures: OCCURS Segment* on page 921.

Syntax: How to Describe a Field in the Master File

```
FIELD[NAME]=field,[ALIAS=]alias,[USAGE=]display,[ACTUAL=]format ,$
```

where:

field

Is a 1- to 48-character field name.

alias

The ALIAS keyword is used for certain fields which require specific ALIAS values.

display

Is the display format for the field.

format

Is the definition of the IDMS/DB field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, ,A2, A2,$
```

ALIAS

The ALIAS keyword is used for certain fields which require specific ALIAS values:

- Database key fields – the ALIAS value is DBKEY.
- ORDER fields – the ALIAS value is ORDER.
- Fields on LRF-based records (LR or ASF) that correspond to the last fields of their underlying physical record types. Filler fields may be required if the last field does not end on a double-word boundary. The ALIAS for each filler field is a unique name with a maximum of eight characters counting the .END suffix. The adapter uses this suffix to correctly address LRF records for LRF calls.
- GROUP fields – an ALIAS is always required or a message results.

The following is an example of a filler field in a segment that describes an LRF record. Suppose a record called JOB-EMPOSITION is defined to the IDMS/DB subschema with seven fields: the first three fields are derived from a physical (DML) record type called JOB; the last four fields, from the EMPOSITION physical record type. The Master File syntax for this LRF record looks like this:

```
FILENAME=JOBMAST, SUFFIX=IDMSR,$
  SEGNAME=JOBEMP, SEGTYPE=S,$
    FIELDNAME=JOBID       ,ALIAS=          ,USAGE=A4   ,ACTUAL=A4  ,$
    FIELDNAME=TITLE       ,ALIAS=          ,USAGE=A20  ,ACTUAL=A20 ,$
    FIELDNAME=DESCRIPTN   ,ALIAS=CMTS.END ,USAGE=A120 ,ACTUAL=A120,$
    FIELDNAME=START_DTE   ,ALIAS=          ,USAGE=A6YMD,ACTUAL=A6  ,$
    FIELDNAME=FINISH_DTE  ,ALIAS=          ,USAGE=A6YMD,ACTUAL=A6  ,$
    FIELDNAME=SALARY_GRADE,ALIAS=          ,USAGE=P4   ,ACTUAL=Z2  ,$
    FIELDNAME=SALARY      ,ALIAS=          ,USAGE=P10.2,ACTUAL=P5  ,$
    FIELDNAME=            ,ALIAS=FILL.END ,USAGE=A1   ,ACTUAL=A8  ,$
```

In this example, the filler field corresponds to the last field of the EMPOSITION record type, because LRF records must lie on a double-word boundary. The IDMS/DB filler field is stored to make the record length a multiple of 8. IDMS/DB filler fields on physical record types underlying LRF views must become filler fields. These filler fields can have any field name or contain a blank; its ALIAS must include the .END suffix. Since the JOB record type is 144 bytes (total of ACTUAL formats), it does not need a filler; the last field DESCRIPTN only requires an alias with an .END suffix.

USAGE

Master Files require you to specify field formats and lengths for USAGE and ACTUAL parameters.

The USAGE and ACTUAL keywords describe the data format of the field. The USAGE parameter defines the field length for the fields. Allow for the maximum possible number of characters or digits including decimal points. You may include valid edit options without increasing the length size.

Note: For network record types, the USAGE and ACTUAL formats of zoned CALC and zoned index fields must be described as alphanumeric (A).

ACTUAL

The ACTUAL parameter defines the field length for COBOL fields found in the IDMS/DB file. The number of internal storage bytes used by COBOL determines the actual length of the field for these formats:

Alphanumeric (A)	Equals the number of characters described in the PICTURE clause.
Zoned decimal (Z)	Equals the number of characters described in the PICTURE clause.
Integer (I)	Equals 2 or 4, corresponding to decimal lengths of 1-4 or 5-9 in the PICTURE clause.
Floating-point (F)	Equals 4 bytes.
Double-precision (D)	Equals 8 bytes.
Packed decimal (P)	Equals (number of PICTURE digits / 2) + 1; excluding sign (S) or implied decimal (V).

Use the following chart as a guide for describing ACTUAL formats:

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	1	P1	P1
COMP-3	S9V99	2	P2	P5.2
COMP-3	9(4)V9(3)	4	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	4	I4	I7

Note:

- For COMP-1 and COMP-2, allow for the maximum possible digits.
- For COBOL DISPLAY fields with zoned decimal, server formats must be packed (P).
- For COMP-1 and COMP-2, PICTURE clauses are not permitted for internal floating-point formats (F and D).

GROUP Fields

The GROUP keyword identifies a set of fields following it with a single name. This GROUP name is any unique name up to 48 characters in length. Its usage is similar to that of a COBOL group name. Generally, this attribute is used for IDMS/DB indexed or CALC fields.

Note:

- USAGE and ACTUAL format types for a GROUP field are always alphanumeric (A).
- USAGE and ACTUAL format lengths for a GROUP field are the sums of the field lengths that form the GROUP field.

For example, a GROUP field called EMP_NAME is composed of two fields, FIRST_NAME and LAST_NAME. Notice the USAGE and ACTUAL formats:

```
GROUP=EMP_NAME    ,ALIAS=  ,USAGE=A25  ,ACTUAL=A25 ,$
  FIELD=FIRST_NAME,ALIAS=  ,USAGE=A10  ,ACTUAL=A10 ,$
  FIELD=LAST_NAME ,ALIAS=  ,USAGE=A15  ,ACTUAL=A15 ,$
```

The FIELDTYPE keyword identifies indexed fields, both SPF and Integrated, on network record types. Omit it for LRF records.

IDMS/DB Database Key

The database key of a network record type corresponding to a segment can optionally be described as the last field in the segment. To specify a database key, use the following values:

Keyword	Description
FIELDNAME	Is any unique key. It can be a maximum of 48 characters.
ALIAS	Value is DBKEY.
USAGE	Value is I10.
ACTUAL	Value is I4.

The database key is used to select records from entry segments when screening on particular IDMS/DB values in the user request.

Remote Descriptions

SEGTYPEs KLU and KL specify unique and non-unique segments for which the aggregate fields are remotely described in another Master File. In other words, a KLU or KL segment contains no field information; its fields are fully defined in another Master File. The CRFILE keyword specifies the source or remote Master File. At execution time, the remote field descriptions are, in effect, copied into the current Master File.

Generally, KLU and KL segment types are used when several Master Files are constructed for the same IDMS/DB subschema. They save typing and maintenance effort. Use KLU if the underlying segment is unique (U); KL if the underlying segment is non-unique (S). Remote descriptions have no logical implications regarding parent/descendant relationships, nor do they impact their implementation.

To specify a remote description, you must give the KLU or KL segment the same name as the source segment in the remote Master File to ensure the proper use of field descriptions. The source file is specified as the CRFILE value. For example:

```
SEGMENT=SALES,PARENT=DEPT,SEGTYPE=KL,CRFILE=RECORDS,$
```

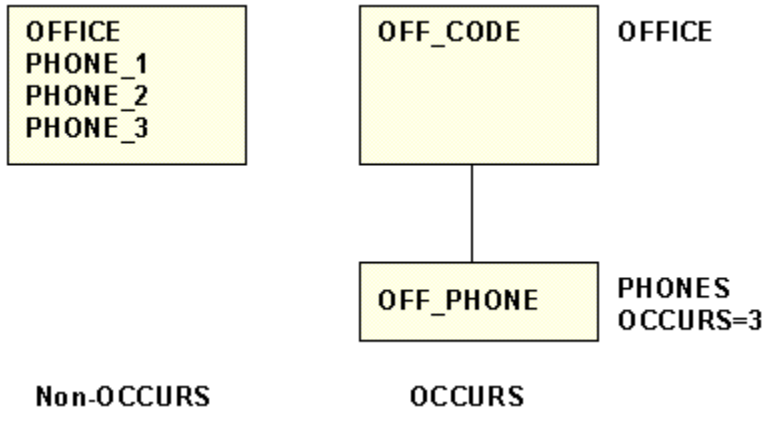
This segment declaration indicates that the field descriptions for the SALES segment are obtained from the SALES segment in the RECORDS Master File. The SEGTYPE for the SALES segment in the RECORDS Master File cannot be KL or KLU; only U or S. Only field names and their attributes from the source file are used; original segment attributes are not. The source file does not have to be a file description as long as it describes the named segment.

Intra-record Structures: OCCURS Segment

A common record structure is one where a field or group of adjacent fields is repeated in the same record type. In COBOL and PL/I syntax, repeating intra-record structures are defined through an OCCURS clause. The server equivalent of an OCCURS clause is an OCCURS segment.

For example, the OFFICE record type contains the field OFFICE-PHONE that occurs three times. The corresponding OFFICE segment can list three separate fields with different field names, or alternately OFFICE-PHONE can be described in a separate descendant OCCURS segment. Using the OCCURS method, each office phone is referenced by a single name (see the Master File for EMPSS01).

The diagram below depicts the non-OCCURS method and the OCCURS method.



OCCURS segments have two attributes or keywords: ORDER and POSITION. The POSITION keyword directs the server to OCCURS segments when non-repeating fields exist between repeating fields. The ORDER keyword creates a fictitious count field that may be decoded.

Describing the Repeating Group to the Server

Any fixed or variable-length record type described in COBOL can be mapped into a server hierarchy using OCCURS segments. A simple OCCURS segment is a descendant of the parent segment which contains non-repeating fields found in the IDMS/DB record type. You must specify the OCCURS keyword on the descendant segment declaration that describes the repeating group. Like the COBOL OCCURS clause, the value of this keyword may be a numeric constant or a field name. The numeric constant indicates a fixed number of repetitions; a field name indicates a count field in the parent segment that maintains a count of the number of occurrences.

OCCURS segments also describe parallel and nested intra-record hierarchical structures. Parallel sets of repeating groups are described as multiple descendant segments of the same parent. In a nested structure, where a repeating group contains another repeating group, one OCCURS segment is the parent of another. Fixed and variable OCCURS segments can be intermixed in any order.

The restrictions for OCCURS segments are as follows:

- The count field for a variably-occurring repeating group must be located physically before the repeating group in the parent of the OCCURS segment.
- A record structure that has a variable number of occurrences but no count field is not within the scope of IDMS/DB.
- The SEGTYPE for an OCCURS segment is specified as S or KL, indicating that it is a non-unique segment.
- OCCURS segments must be defined in the Master File in the same order as they appear in the actual record type, unless the POSITION attribute is used.
- OCCURS segments do not have a corresponding segment declaration in the Access File, because the Adapter for IDMS/DB simulates them using the parent record. OCCURS segments do not generate any additional IDMS/DB calls.

From the server standpoint, OCCURS segments are indistinguishable from other segments. The server processes them, if referenced, in the usual top-to-bottom, left-to-right retrieval order.

For example, consider this COBOL budget record type:

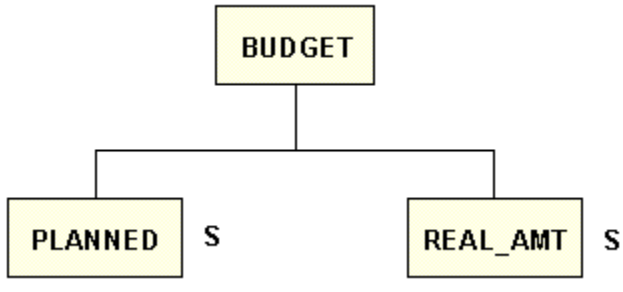
```
01  BUDGET-RCD.
02  ACCOUNT           PIC XXX.
02  ACTUAL-COUNT      PIC 99.
02  PLANNED-AMT       PIC 9(9) OCCURS 12 TIMES.
02  ACTUAL-AMT        PIC 9(9) OCCURS 12 TIMES
                       DEPENDING ON ACTUAL-COUNT.
```

The equivalent Master File is:

```
SEGMENT=BUDGET,SEGTYPE=S,$
   FIELD=ACCOUNT      ,ALIAS=   ,USAGE=A3  ,ACTUAL=A3       ,$
   FIELD=ACTUAL_COUNT ,ALIAS=   ,USAGE=P4  ,ACTUAL=Z2       ,$

SEGMENT=PLANNED ,PARENT=BUDGET,SEGTYPE=S,OCCURS=12          ,$
   FIELD=PLANNED_AMT  ,ALIAS=   ,USAGE=P12 ,ACTUAL=Z9       ,$

SEGMENT=REAL_AMT,PARENT=BUDGET,SEGTYPE=S,OCCURS=ACTUAL_COUNT,$
   FIELD=ACTUAL_AMT   ,ALIAS=   ,USAGE=P12 ,ACTUAL=Z9       ,$
```

The mapping principle is very simple: the non-repeating field in the record type is described in one parent segment, and the parallel COBOL OCCURS structures are made into descendant OCCURS segments. The OCCURS keyword on the descendant segments specifies either a number (fixed occurrences) or a count field (variable occurrences). In this case, the count field ACTUAL_COUNT is located in the immediate parent segment called BUDGET and is specified on the REAL_AMT descendant segment.

The following is a diagram for this Master File example:



If the PLANNED or REAL_AMT segments had repeating structures, they would in turn be parents of OCCURS segments defined by the same principles. The BUDGET segment could have other non-OCCURS descendants. The PLANNED and REAL_AMT segments might have CALC-based or index-based descendants. However, set-based descendants of this record type would be tied to the BUDGET segment, not to the PLANNED or REAL_AMT segments.

POSITION

OCCURS segments are defined in the same order as they appear in the actual record type. In some cases, COBOL OCCURS clauses are separated by non-repeating fields. The POSITION keyword in a Master File indicates that the repeating fields are located in the middle of non-repeating fields.

The POSITION attribute can only be used for a repeating group with a fixed number of occurrences. This means that the OCCURS attribute of the descendant segment must equal a numeric constant and not a count field.

Suppose the previous COBOL record type looked like this:

```
01  BUDGET-RCD.
02  ACCOUNT         PIC XXX.
02  PLANNED-AMT     PIC 9(9) OCCURS 12 TIMES.
02  ACTUAL-COUNT    PIC 99.
02  ACTUAL-AMT      PIC 9(9) OCCURS 12 TIMES
                    DEPENDING ON ACTUAL-COUNT.
```

Here, the two repeating structures PLANNED-AMT and ACTUAL-AMT are separated by the non-repeating field ACTUAL-COUNT, which clearly belongs to the BUDGET segment. You must indicate in the Master File that the first occurrence of the PLANNED segment will not immediately follow the ACCOUNT field in the BUDGET segment (the PLANNED-AMT field is described in a separate descendant segment). The POSITION keyword accomplishes this task by directing the server to the descendant segment named PLANNED.

The corresponding Master File looks like this:

```
SEGMENT=BUDGET,SEGTYPE=S,$
  FIELD=ACCOUNT      ,ALIAS=   ,USAGE=A3   ,ACTUAL=A3        ,$
  FIELD=PLANNED_SEG1 ,ALIAS=   ,USAGE=A108 ,ACTUAL=A108      ,$
  FIELD=ACTUAL_COUNT ,ALIAS=   ,USAGE=P4   ,ACTUAL=Z2        ,$


SEGMENT=PLANNED ,PARENT=BUDGET,SEGTYPE=S,OCCURS=12,
        POSITION=PLANNED_SEG1                               ,$
  FIELD=PLANNED_AMT  ,ALIAS=   ,USAGE=P12  ,ACTUAL=Z9        ,$


SEGMENT=REAL_AMT,PARENT=BUDGET,SEGTYPE=S,OCCURS=ACTUAL_COUNT ,$
  FIELD=ACTUAL_AMT   ,ALIAS=   ,USAGE=P12  ,ACTUAL=Z9        ,$
```

The POSITION keyword in the PLANNED segment names a field called PLANNED_SEG1 in BUDGET, the immediate parent segment. PLANNED_SEG1 in the parent coincides with the first field of the first occurrence in PLANNED, the OCCURS segment. The REAL_AMT segment does not require a POSITION keyword, because the position of its first occurrence is correctly inferred as following the last described field in the BUDGET segment.

In the previous example, the PLANNED_SEG1 spans all occurrences of the PLANNED segment. As an alternative, 12 individual fields named PLANNED_SEG1 through PLANNED_SEG12 could be described instead in the BUDGET segment. Each individual field would need the appropriate numeric format. Then you could refer to any one of the 12 amount fields by its specific name, or generically through the PLANNED_AMT field. The POSITION attribute can always be used for this purpose, even when it is not required for positioning the first position of an OCCURS segment located in the middle of the record type.

ORDER Field

Sometimes the sequence of fields within an OCCURS segment is significant. For example, each instance of the repeating field may represent one quarter of the year, but the segment may not have a field that specifies the quarter to which it applies.

ORDER is an optional counter used to identify the sequence number within a group of repeating fields. Specify it when the order of data is important. The ORDER field does not represent an existing field in the data source; it is used only for internal processing.

Syntax: **How to Identify an ORDER Field**

The ORDER field must be the last field described in the OCCURS segment

```
FIELDNAME=name, ALIAS=ORDER, USAGE=In, ACTUAL=I4 ,$
```

where:

name

Is any valid field name.

I*n*

Is an integer format.

Note:

- The ALIAS value must be ORDER.
- The ACTUAL format must be I4.
- The ORDER field must be the last field defined in the OCCURS segment.

Example: **Using the ORDER Field in Requests**

In requests, you can use the value of the ORDER field. You can also specify a DEFINE statement in the Master File to translate it to more meaningful values. For example:

```
DEFINE QTR/A3 = DECODE ORDER(1 '1ST' 2 '2ND' 3 '3RD' 4 '4TH');
```

A subsequent request could include:

```
SELECT TOT.TAXES FROM JOBMAST
WHERE QTR = 1
```

or

```
SELECT QTR,BALANCE,INTEREST
```

Access Files for IDMS/DB

An Access File describes the database access information, for example, database number, file number, and descriptor fields.

Note that before you can access a CA-IDMS/DB file, you must first describe it to the server in two files: a Master File, and an associated Access File. A Master File is required to describe the CA-IDMS/DB file in standard server terminology to the Adapter for IDMS/DB. For more information on the Master File, see *Managing IDMS/DB Metadata* on page 910.

An Access File is required to translate a request for network record types and LRF records into the appropriate CA-IDMS/DB Data Management Language (DML) retrieval commands. This file description consists of 80-character records called declarations in comma-delimited format (keyword=value).

An Access File contains three kinds of declarations:

- Subschema
- Segment
- Index

Subschema declarations identify the subschema used, the IDMS/DB release under which the subschema was compiled, the calling mode to be used to retrieve records, and whether a trace is to be produced. Several subschema declarations may be specified in a single Access File. Each subschema declaration is followed by its segment and index declarations. *Subschema Declaration Keywords* on page 927 describes keywords for subschema declarations.

Access File segment declarations indicate the IDMS/DB record information and the parent/descendant relationship for each network record type or LRF record described as a segment in a Master File (Access File segment declarations are not defined for OCCURS segments). Segment declarations can be specified in any order after their corresponding subschema declaration. *Segment Declaration Keywords for Network Record Types* on page 929 and *Segment Declaration Keywords for LRF Records* on page 932 describe segment declarations for network record types and LRF records, respectively.

Index declarations provide information about each IDMS/DB index. They may also be in any order, one for each indexed field described in the Master File, after their corresponding subschema declaration. *Index Declaration Keywords for Network Record Types* on page 934 describes index declarations for network record types.

Access File Syntax

Each declaration consists of a list of keyword and value pairs, separated by commas. The list is free-form and may span several lines; keywords may be specified in any order. Each declaration is completed with a comma followed by a dollar sign ($). For example, this Access File contains three declarations:

```
SSCHEMA=PAYROLL,RELEASE=15,INDEXAREA=PRIMARY-IX-AREA,
    DBNAME=EMPDEMO,DICTNAME=APPLDICT,$

SEGNAME=ACCOUNT,RECORD=PAYREC,AREA=PAY-REGION,
    CLCFLD=EMPLOYEE_ID,CLCDUP=N,$

IXSET=IXREC-SSN,IXAREA=IX-AREA1,IXFLD=PERS_SSN,
    IXDUP=N,IXORD=A,$
```

Blank lines and lines starting with an asterisk (*) in column 1 are treated as comments. Leading and trailing blanks around keywords and values are ignored. Values that contain commas, equal signs, dollar signs, or spaces must be enclosed in single quotation marks.

Subschema Declaration Keywords

If your Master File defines record types and LRF records from multiple IDMS/DB subschemas, the Access File should contain multiple subschema declarations. After each subschema declaration, list its segment and index declarations.

Subschema declarations for DML and LRF subschemas contain the following keywords; certain keywords are optional as explained in the following summary chart.

Keyword	Description
SSCHEMA	IDMS/DB subschema name.
RELEASE	Release of the IDMS/DB software that was used in the last compilation of the subschema. The value can be 10.2, 12, or 12.*x* (where *x* is your release version), 14, or 15.
MODE	Type of IDMS/DB access the adapter is to perform. Specify LR for LR and ASF records; DML is the default value.

Keyword	Description
TRACE	Optional keyword used for debugging purposes. It specifies whether a basic trace of all IDMS/DB calls or a detailed trace of all the parameters passed to IDMS/DB will be displayed. Values can be YES, PARMS, or NO (NO is the default value).
READY	Optional keyword that specifies when an LRF record is built from two or more physical record types located in several database areas. The adapter prepares or readies all the areas of the subschema. Values can be ALL or null or omitted entirely.
DBNAME	IDMS/DB database name from the DBNAME table corresponding to its subschema. It can be used in local or Central Version mode to translate the subschema name into the proper load module(s) for data access.
INDEXAREA	Name of the primary SPF index area. Required for any subschema with SPF indices.

CV Mode Only

Keyword	Description
DICTNAME	Secondary dictionary load area, if the subschema is not located in the primary dictionary or in a load PDS. Remember that ASF subschemas are located in secondary dictionary load areas by default, so if your Access File describes an ASF record, you must specify this keyword.
NODE	Distributed Database Systems (DDS) node location of an IDMS/DB distributed database. The value is the IDMS/DB data dictionary table entry that identifies the DDS node location of an IDMS/DB distributed database. This keyword is required only if DDS is installed at a user site and if the subschema is located in a remote site location.

Keyword	Description
DICTNODE	DDS node location of an IDMS/DB distributed database subschema in a secondary dictionary load area. The value is the IDMS/DB data dictionary table entry that identifies the DDS node location of an IDMS/DB distributed database subschema. This keyword is required only if DDS is installed at the user site and if the subschema is located in a remote site location.

Note: When running using DDS, Central Version must be used. DDS access is not supported in local mode.

Segment Declaration Keywords for Network Record Types

Your use of keywords in segment declarations for DML record types depends on whether the record type contains a CALC key, acts as a descendant segment, or contains an index. The following keywords are common to all segment declarations.

Keyword	Description
SEGNAM	Corresponding Master File segment name of the DML record type.
RECORD	IDMS/DB record type name.
AREA	IDMS/DB area name that contains the record type.

If your record type is CALC (that is, if the record contains a CALC key) include the two keywords below. These keywords are required for all CALC record types, regardless of how their parent/descendant relationships are implemented.

Keyword	Description
CLCFLD	Master File field name of the CALC field.
CLCDUP	Indicates if the CALC field allows duplicates. The value can be Y or N.

Record types are assigned parent/descendant relationships in the server. These relationships are based on sets and CALC or index fields. Keywords for descendant segments follow. Consult your Master File to determine whether a segment is a descendant or parent.

Field	Keyword	Description
	ACCESS	Indicates the relationship that exists between record types. The value CLC or IX specifies an embedded cross-reference based on a CALC or indexed field. The value SET indicates a physical relationship based on a set of pointers.
Set-Based (ACCESS=SET)	SETNAME	Name of the set relating a descendant to its parent.
	SETMBR	Specifies whether the set membership is mandatory/automatic, mandatory/manual, optional/automatic, or optional/manual. This information is used to verify set membership at execution time. To determine the appropriate membership value, check the set label on your Bachman diagram. Values can be MA, MM, OA, or OM.
	GETOWN	Allows or inhibits GET OWNER calls which obtain the owner records from a member record type. If the value is Y, the adapter issues GET OWNER calls to retrieve the owner record in the set when SEGTYPE is U or KLU. If the value is N, GET OWNER calls are inhibited. Specify N only when the set has no owner pointers and long member record chains are apt to occur. When GET OWNER calls are inhibited, the owner record type cannot be a descendant of its member. In other words, if GET OWNER calls are inhibited, SEGTYPE cannot be U or KLU.
	MULTMBR	Indicates whether the set, in which this record type participates, contains more than one member record type. Values can be Y or N.
	KEYFLD	Server field that sequences the set. Select a field from a parent or descendant segment as the value of KEYFLD in the descendant segment declaration.

Field	Keyword	Description
	SETORD	A (ascending) or D (descending) for sorted set sequence. This keyword is required.
	SETDUP	Y or N if duplicates are allowed. Required for sorted sets.
CALC-Based (ACCESS=CLC)	KEYFLD	Provides the search value to read CALC and index fields in descendant segments. These search values are located in parent segment fields. Specify the parent field name for the value of KEYFLD in the descendant segment declaration. The KEYFLD keyword is especially important when the two record types in a parent/descendant relationship are from different subschemas. The record type that acts as the descendant segment is the entry point into the second subschema. It must have a CALC key (CLCFLD) or index set (SETNAME) with ACCESS=CLC or ACCESS=IX, respectively. The descendant segment declaration must also list the KEYFLD value from the parent segment in the first subschema.
Index-Based (ACCESS=IX)	KEYFLD	See above.
	SETNAME	IDMS/DB name of the index set. A corresponding index declaration is required.

Example: Describing One Subschema With Two Segments

This example shows one subschema with two segments that are CALC record types. The INVOICE record type has a set-based relationship with the CUSTOMER record type.

```
SSCHEMA=SAMPSSCH,RELEASE=15,
SEGNAM=CUSTOMER,RECORD=CUSTOMER,AREA=CUSTOMER-REGION,
 CLCFLD=CUST_NUMBER,CLCDUP=N,$
SEGNAM=INVOICE,RECORD=INVOICE,AREA=INVO-REGION,
 CLCFLD=INV_NUMBER,CLCDUP=N,ACCESS=SET,
 SETNAME=CUSTOMER-INVO,SETMBR=MA,GETOWN=Y,MULTMBR=N,$
```

Example: Describing Two Subschemas

The following example shows two subschemas. The INSURANCE-PLAN record type has a CALC-based relationship with the COVERAGE record type.

```
SSCHEMA=EMPSS01,RELEASE=15,
SEGNAM=COVERAGE,RECORD=COVERAGE,AREA=INS-DEMO-REGION,$

SSCHEMA=EMPSS03,RELEASE=15,$
SEGNAM=INSURNCE,RECORD=INSURANCE-PLAN,
 AREA=INS-DEMO-REGION,CLCFLD=INS_PLAN_CODE,CLCDUP=N,
 ACCESS=CLC,KEYFLD=COV_CODE,$
```

If your record type contains an indexed field, you may suppress area sweeps when the segment is used as a point of entry into the database. To prevent area sweeps, specify the optional keyword below on the segment declaration. Only those record instances connected to the specified index field are accessed.

Keyword	Description
SEQFIELD	Master File field name (FIELDTYPE=I) of the index.

Note: This optional keyword requires an index declaration (see *Index Declaration Keywords for Network Record Types* on page 934).

Segment Declaration Keywords for LRF Records

Segment declaration keywords for LR and ASF records are basically the same as those for network (DML) record types. Specified values, of course, differ.

Keyword	Description
SEGNAM	Corresponding Master File segment name of the LRF record.
RECORD	IDMS/DB name of the LRF record.
LR	Value is Y.
AREA	IDMS/DB area name that contains physical record types. Specify READY=ALL on the subschema declaration for more than one area (if fields originate from many areas).

LRF records use embedded cross-references to create parent/descendant relationships. Specify the following keywords for LRF records that act as descendant segments:

Keyword	Description
ACCESS	Value is LR.
KEYFLD	Master File field name found in the parent.
IXFLD	Master File field name found in the descendant.

The KEYFLD and IXFLD keywords are required to implement parent/descendant relationships. The KEYFLD keyword specifies a field from the parent segment that provides search values. The value of IXFLD, in turn, is a field in the descendant segment that contains equivalent values for KEYFLD. Any field may be selected for IXFLD provided that the record possesses a null SELECT clause.

Suppose your Master File for one subschema contained two LRF records that act as parent and descendant segments:

```
FILE=DEPTEMP,SUFFIX=IDMSR,$
SEGNAME=DEPTEMP,$
 .
 .
 .
 FIELD=EMP_ID      ,ALIAS=   ,USAGE=A4 ,ACTUAL=A4,$
SEGNAME=EMPJOB,  PARENT=DEPTEMP,  SEGTYPE=S,$
 FIELD=EMPLOYEE_ID ,ALIAS=   ,USAGE=A4 ,ACTUAL=A4,$
 .
 .
 .
```

The common field is EMPLOYEE_ID; its field format is the same in both segments. The EMPJOB segment is the descendant of DEPTEMP. The descendant segment declaration in the Access File below contains KEYFLD and IXFLD values:

```
SSCHEMA=EMPSS06,RELEASE=15,MODE=LR,READY=ALL
  DBNAME=EMPDEMO,DICTNAME=APPLDICT,$

SEGNAM=DEPTEMP,RECORD=DEPT-EMPLOYEE,
  AREA=EMP-DEMO-REGION,LR=Y,$

SEGNAM=EMPJOB,RECORD=EMPLOYEE-JOB,
  AREA=ORG-DEMO-REGION,LR=Y,
  ACCESS=LR,KEYFLD=EMP_ID,IXFLD=EMPLOYEE_ID,$
```

If the EMPJOB segment (EMPLOYEE-JOB record) belonged to a different subschema, the Access File example above would include another subschema declaration positioned between the two segment declarations.

Index Declaration Keywords for Network Record Types

The Adapter for IDMS/DB supports two indexing schemes: the traditional method of indexing, using the Sequential Processing Facility (SPF), and IDMS/DB Integrated Indexes. Under SPF, index entries are stored in separate index areas. As of IDMS/DB Release 10, indexes may be integrated with the Database Management System (DBMS).

The following keywords apply to declarations for both SPF and Integrated Indexes. For an example of an index declaration, see *Access File Syntax* on page 927.

Keyword	Description
IXSET	IDMS/DB setname of the index set.
IXFLD	Corresponding Master File field name with FIELDTYPE=I.
IXDUP	Indicates if duplicate index values are allowed. Value can be Y or N.
IXORD	Indicates the sort order of the index. Value can be A (ascending) or D (descending).
IXAREA	IDMS/DB area name of the index; usage varies, see below.

Note:

- This declaration is not used in LRF Access Files.
- For subschemas with SPF indexing, the IXAREA keyword is required.
- For subschemas with Integrated Indexes, the IXAREA keyword is omitted unless the index entries reside in a different area from the record type being indexed.
- A non-unique descendant with an SPF index may not have descendants (immediate or several levels removed) that are related to the same index. This is due to the lack of an IDMS/DB command, which would restore currency after it has been disturbed by the record retrieval of a descendant segment. This restriction does not apply to SPF unique descendants or Integrated Index descendants.

IDMS/DB Sample File Descriptions

The following sections contain:

- The *Schema: EMPSCHM*.
- A *Network Subschema: EMPSS01* for the schema EMPSCHM, plus corresponding *Master File for Network* and *Access File for Network*.
- An *LRF Subschema: EMPSS02* for the schema EMPSCHM, plus corresponding *Master File for LRF* and *Access File for LRF*.
- A sample of a NULL SELECT clause that creates an *Sample of a Partial LRF Record*.
- A sample from a subschema that contains *SPF Indexes*.

Note: Some samples are annotated to illustrate specific clauses.

Schema: EMPSCHM

This schema is the physical description of the IDMS/DB EMPSCHM database. It contains the following items:

- Area-to-file and area-to-DDNAME mapping.
- A CALC-based record.
- A VIA-based record.
- A sorted set ordered by the SKILL-LEVEL field.
- A sorted set ordered by Integrated Indexes using the SKILL-LEVEL field.
- An Integrated Index set ordered by the SKILL-NAME field.

```
ADD SCHEMA NAME IS EMPSCHM VERSION IS 1

SCHEMA DESCRIPTION IS 'EMPLOYEE DEMO DATABASE'

COMMENTS 'INSTALLATION: COMMONWEATHER CORPORATION'
    .

       ADD FILE NAME IS EMPDEMO ASSIGN TO EMPDEMO
           DEVICE TYPE IS 3380
    .

       ADD FILE NAME IS INSDEMO ASSIGN TO INSDEMO
                  DEVICE TYPE IS 3380
    .
```

```
   ADD FILE NAME IS ORGDEMO ASSIGN TO ORGDEMO
             DEVICE TYPE IS 3380
.

   ADD AREA NAME IS EMP-DEMO-REGION
             RANGE IS 75001 THRU 75100
             WITHIN FILE EMPDEMO FROM 1 THRU 100
.

   ADD AREA NAME IS ORG-DEMO-REGION
             RANGE IS 75151 THRU 75200
             WITHIN FILE ORGDEMO FROM 1 THRU 50
.

   ADD AREA NAME IS INS-DEMO-REGION
             RANGE IS 75101 THRU 75150
             WITHIN FILE INSDEMO FROM 1 THRU 50
.

   ADD RECORD NAME IS COVERAGE
       SHARE STRUCTURE OF RECORD COVERAGE VERSION IS 1
       RECORD ID IS 0400
       LOCATION MODE IS VIA             EMP-COVERAGE SET
       WITHIN AREA INS-DEMO-REGION
       OFFSET  2 PAGES FOR 48 PAGES
.

   ADD RECORD NAME IS DENTAL-CLAIM
       SHARE STRUCTURE OF RECORD DENTAL-CLAIM VERSION IS 1
       RECORD ID IS 0405
       LOCATION MODE IS VIA          COVERAGE-CLAIMS SET
       WITHIN AREA INS-DEMO-REGION
       OFFSET  2 PAGES FOR 48 PAGES
       MINIMUM ROOT LENGTH IS             130 CHARACTERS
       MINIMUM FRAGMENT LENGTH IS          RECORD LENGTH
.

   ADD RECORD NAME IS DEPARTMENT
       SHARE STRUCTURE OF RECORD DEPARTMENT VERSION IS 1
       RECORD ID IS 0410
       LOCATION MODE IS CALC          USING DEPT-ID-0410
                                  DUPLICATES NOT ALLOWED
       WITHIN AREA ORG-DEMO-REGION
       OFFSET  2 PAGES FOR 48 PAGES
.
```

```
   ADD RECORD NAME IS EMPLOYEE
       SHARE STRUCTURE OF RECORD EMPLOYEE VERSION IS 1
       RECORD ID IS 0415
       LOCATION MODE IS CALC           USING EMP-ID-0415
                                 DUPLICATES NOT ALLOWED
       WITHIN AREA EMP-DEMO-REGION
       OFFSET  2 PAGES FOR 98 PAGES
.

   ADD RECORD NAME IS EMPOSITION
       SHARE STRUCTURE OF RECORD EMPOSITION VERSION IS 1
       RECORD ID IS 0420
       LOCATION MODE IS VIA           EMP-EMPOSITION SET
       WITHIN AREA EMP-DEMO-REGION
       OFFSET  2 PAGES FOR 98 PAGES
.

   ADD RECORD NAME IS EXPERTISE
       SHARE STRUCTURE OF RECORD EXPERTISE VERSION IS 1
       RECORD ID IS 0425
       LOCATION MODE IS VIA            EMP-EXPERTISE SET
       WITHIN AREA EMP-DEMO-REGION
       OFFSET  2 PAGES FOR 98 PAGES
.

   ADD RECORD NAME IS HOSPITAL-CLAIM
       SHARE STRUCTURE OF RECORD HOSPITAL-CLAIM VERSION IS 1
       RECORD ID IS 0430
       LOCATION MODE IS VIA          COVERAGE-CLAIMS SET
       WITHIN AREA INS-DEMO-REGION
       OFFSET  2 PAGES FOR 48 PAGES
.

   ADD RECORD NAME IS INSURANCE-PLAN
       SHARE STRUCTURE OF RECORD INSURANCE-PLAN VERSION IS 1|
       RECORD ID IS 0435
       LOCATION MODE IS CALC    USING INS-PLAN-CODE-0435
                                  DUPLICATES NOT ALLOWED
       WITHIN AREA INS-DEMO-REGION
       OFFSET  1 PAGE  FOR  1 PAGE
.
```

```
   ADD RECORD NAME IS JOB
       SHARE STRUCTURE OF RECORD JOB VERSION IS 1
       RECORD ID IS 0440
       LOCATION MODE IS CALC           USING JOB-ID-0440
                                   DUPLICATES NOT ALLOWED
       WITHIN AREA ORG-DEMO-REGION
       OFFSET  2 PAGES FOR 48 PAGES
       MINIMUM ROOT LENGTH IS CONTROL LENGTH
       MINIMUM FRAGMENT LENGTH IS RECORD LENGTH
       CALL IDMSCOMP BEFORE STORE
       CALL IDMSCOMP BEFORE MODIFY
       CALL IDMSDCOM AFTER GET
.

   ADD RECORD NAME IS NON-HOSP-CLAIM
       SHARE STRUCTURE OF RECORD NON-HOSP-CLAIM VERSION IS 1
       RECORD ID IS 0445
       LOCATION MODE IS VIA          COVERAGE-CLAIMS SET
       WITHIN AREA INS-DEMO-REGION
       OFFSET 2 PAGES FOR 48 PAGES
       MINIMUM ROOT LENGTH IS             248 CHARACTERS
       MINIMUM FRAGMENT LENGTH IS          RECORD LENGTH
.

   ADD RECORD NAME IS OFFICE
       SHARE STRUCTURE OF RECORD OFFICE VERSION IS 1
       RECORD ID IS 0450
       LOCATION MODE IS CALC      USING OFFICE-CODE-0450
                                  DUPLICATES NOT ALLOWED
       WITHIN AREA ORG-DEMO-REGION
       OFFSET 2 PAGES FOR 48 PAGES
.

   ADD RECORD NAME IS SKILL
       SHARE STRUCTURE OF RECORD SKILL VERSION IS 1
       RECORD ID IS 0455
       LOCATION MODE IS CALC         USING SKILL-ID-0455
                                   DUPLICATES NOT ALLOWED
       WITHIN AREA ORG-DEMO-REGION
       OFFSET 2 PAGES FOR 48 PAGES
.

   ADD RECORD NAME IS STRUCTURE
       SHARE STRUCTURE OF RECORD STRUCTURE VERSION IS 1
       RECORD ID IS 0460
       LOCATION MODE IS VIA                  MANAGES SET
       WITHIN AREA EMP-DEMO-REGION
       OFFSET 2 PAGES FOR 98 PAGES
.
```

```
ADD SET NAME IS COVERAGE-CLAIMS
    ORDER IS LAST
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS COVERAGE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
    MEMBER IS HOSPITAL-CLAIM
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
        MANDATORY AUTOMATIC
    MEMBER IS NON-HOSP-CLAIM
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
        MANDATORY AUTOMATIC
    MEMBER IS DENTAL-CLAIM
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
        MANDATORY AUTOMATIC
.

ADD SET NAME IS DEPT-EMPLOYEE
    ORDER IS SORTED
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS DEPARTMENT
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
    MEMBER IS EMPLOYEE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
        LINKED TO OWNER
            OWNER DBKEY POSITION IS AUTO
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( EMP-LAST-NAME-0415
                           EMP-FIRST-NAME-0415 )
            DUPLICATES LAST
.

ADD SET NAME IS EMP-COVERAGE
    ORDER IS FIRST
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYEE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
    MEMBER IS COVERAGE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
        LINKED TO OWNER
            OWNER DBKEY POSITION IS AUTO
        MANDATORY AUTOMATIC
.
```

```
   ADD SET NAME IS EMP-EMPOSITION
       ORDER IS FIRST
       MODE IS CHAIN LINKED TO PRIOR
       OWNER IS EMPLOYEE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS EMPOSITION
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           MANDATORY AUTOMATIC
.

   ADD SET NAME IS EMP-EXPERTISE
       ORDER IS SORTED
       MODE IS CHAIN LINKED TO PRIOR
       OWNER IS EMPLOYEE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS EXPERTISE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           MANDATORY AUTOMATIC
           DESCENDING KEY IS (SKILL-LEVEL-0425 )
               DUPLICATES FIRST
.

   ADD SET NAME IS EMP-NAME-NDX
       ORDER IS SORTED
       MODE IS INDEX BLOCK CONTAINS 40 KEYS
       OWNER IS SYSTEM
           WITHIN AREA EMP-DEMO-REGION
           OFFSET  1 PAGE  FOR  1 PAGE
       MEMBER IS EMPLOYEE
           INDEX DBKEY POSITION IS AUTO
           OPTIONAL AUTOMATIC
           ASCENDING KEY IS ( EMP-LAST-NAME-0415
                              EMP-FIRST-NAME-0415 )
               COMPRESSED
               DUPLICATES LAST
.
```

```
   ADD SET NAME IS JOB-EMPOSITION
       ORDER IS NEXT
       MODE IS CHAIN LINKED TO PRIOR
       OWNER IS JOB
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS EMPOSITION
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           OPTIONAL MANUAL
.

   ADD SET NAME IS JOB-TITLE-NDX
       ORDER IS SORTED
       MODE IS INDEX BLOCK CONTAINS 30 KEYS
       OWNER IS SYSTEM
           WITHIN AREA ORG-DEMO-REGION
           OFFSET  1 PAGE  FOR  1 PAGE
       MEMBER IS JOB
           INDEX DBKEY POSITION IS AUTO
           OPTIONAL AUTOMATIC
           ASCENDING KEY IS ( TITLE-0440 )
               DUPLICATES NOT ALLOWED
.

   ADD SET NAME IS MANAGES
       ORDER IS NEXT
       MODE IS CHAIN LINKED TO PRIOR
       OWNER IS EMPLOYEE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
       MEMBER IS STRUCTURE
           NEXT DBKEY POSITION IS AUTO
           PRIOR DBKEY POSITION IS AUTO
           LINKED TO OWNER
               OWNER DBKEY POSITION IS AUTO
           MANDATORY AUTOMATIC
.
```

```
ADD SET NAME IS OFFICE-EMPLOYEE
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 30 KEYS
    OWNER IS OFFICE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
    MEMBER IS EMPLOYEE
        INDEX DBKEY POSITION IS AUTO
        LINKED TO OWNER
            OWNER DBKEY POSITION IS AUTO
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( EMP-LAST-NAME-0415
                           EMP-FIRST-NAME-0415 )
            COMPRESSED
            DUPLICATES LAST
.

ADD SET NAME IS REPORTS-TO
    ORDER IS NEXT
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
    MEMBER IS STRUCTURE
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
        LINKED TO OWNER
            OWNER DBKEY POSITION IS AUTO
        OPTIONAL MANUAL
.

ADD SET NAME IS SKILL-EXPERTISE
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 30 KEYS
    OWNER IS SKILL
        NEXT DBKEY POSITION IS AUTO
        PRIOR DBKEY POSITION IS AUTO
    MEMBER IS EXPERTISE
        INDEX DBKEY POSITION IS AUTO
        LINKED TO OWNER
            OWNER DBKEY POSITION IS AUTO
        MANDATORY AUTOMATIC
        DESCENDING KEY IS ( SKILL-LEVEL-0425 )
            DUPLICATES FIRST
.
```

```
    ADD SET NAME IS SKILL-NAME-NDX
        ORDER IS SORTED
        MODE IS INDEX BLOCK CONTAINS 30 KEYS
        OWNER IS SYSTEM
            WITHIN AREA ORG-DEMO-REGION
            OFFSET  1 PAGE  FOR  1 PAGE
        MEMBER IS SKILL
            INDEX DBKEY POSITION IS AUTO
            OPTIONAL AUTOMATIC
            ASCENDING KEY IS ( SKILL-NAME-0455 )
                DUPLICATES NOT ALLOWED
.
    VALIDATE
.
```

Network Subschema: EMPSS01

This subschema shows the network view of the schema EMPSCHM:

```
ADD SUBSCHEMA NAME IS EMPSS01
  OF SCHEMA NAME IS EMPSCHM VERSION  1
DMCL NAME IS EMPDMCL
  OF SCHEMA NAME IS EMPSCHM VERSION  1
COMMENTS 'THIS IS THE COMPLETE VIEW OF EMPSCHM'.
ADD AREA NAME IS EMP-DEMO-REGION.
ADD AREA NAME IS INS-DEMO-REGION.
ADD AREA NAME IS ORG-DEMO-REGION.
ADD RECORD NAME IS COVERAGE.
ADD RECORD NAME IS DENTAL-CLAIM.
ADD RECORD NAME IS DEPARTMENT.
ADD RECORD NAME IS EMPLOYEE.
ADD RECORD NAME IS EMPOSITION.
ADD RECORD NAME IS EXPERTISE.
ADD RECORD NAME IS HOSPITAL-CLAIM.
ADD RECORD NAME IS INSURANCE-PLAN.
ADD RECORD NAME IS JOB.
ADD RECORD NAME IS NON-HOSP-CLAIM.
ADD RECORD NAME IS OFFICE.
ADD RECORD NAME IS SKILL.
ADD RECORD NAME IS STRUCTURE.
ADD SET COVERAGE-CLAIMS.
ADD SET DEPT-EMPLOYEE.
ADD SET EMP-COVERAGE.
ADD SET EMP-EXPERTISE.
ADD SET EMP-NAME-NDX.
ADD SET EMP-EMPOSITION.
ADD SET JOB-EMPOSITION.
ADD SET JOB-TITLE-NDX.
ADD SET MANAGES.
ADD SET OFFICE-EMPLOYEE.
ADD SET REPORTS-TO.
ADD SET SKILL-EXPERTISE.
ADD SET SKILL-NAME-NDX.
GENERATE.
```

Master File for Network

This Master File corresponds to the network subschema EMPSS01:

```
FILE=EMPFULL,SUFFIX=IDMSR ,$
SEGNAME=DEPT,$
  FIELDNAME=DEPT_ID     ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=DEPT_NAME   ,ALIAS=      ,USAGE=A45  ,ACTUAL=A45      ,$
  FIELDNAME=DEPT_HEAD   ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=DEPT_DBKEY  ,ALIAS=DBKEY ,USAGE=I10  ,ACTUAL=I4       ,$
```

```
SEGNAME=EMPLOYE,PARENT=DEPT,SEGTYPE=S,$
  FIELDNAME=EMP_ID      ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4       ,$
  GROUP=EMP_NAME        ,ALIAS=      ,USAGE=A25   ,ACTUAL=A25      ,$
    FIELDNAME=FIRST_NAME,ALIAS=      ,USAGE=A10   ,ACTUAL=A10      ,$
    FIELDNAME=LAST_NAME ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=EMP_STREET  ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20      ,$
  FIELDNAME=EMP_CITY    ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=EMP_STATE   ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  GROUP=EMP_FULL_ZIP    ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9       ,$
    FIELDNAME=EMP_ZIP   ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5       ,$
    FIELDNAME=EMP_ZIP_L ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=EMP_PHONE   ,ALIAS=      ,USAGE=A10   ,ACTUAL=A10      ,$
  FIELDNAME=STATUS      ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  FIELDNAME=SOC_SEC_NUM ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9       ,$
  FIELDNAME=EMP_STRT_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=EMP_TERM_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=EMP_BRTH_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=EMP_DBKEY   ,ALIAS=DBKEY ,USAGE=I10   ,ACTUAL=I4       ,$

SEGNAME=OFFICE,PARENT=EMPLOYE,SEGTYPE=U,$
  FIELDNAME=OFF_CODE    ,ALIAS=      ,USAGE=A3    ,ACTUAL=A3       ,$
  FIELDNAME=OFF_STREET  ,ALIAS=      ,USAGE=A20   ,ACTUAL=A20      ,$
  FIELDNAME=OFF_CITY    ,ALIAS=      ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=OFF_STATE   ,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  GROUP=OFF_FULL_ZIP    ,ALIAS=      ,USAGE=A9    ,ACTUAL=A9       ,$
    FIELDNAME=OFF_ZIP   ,ALIAS=      ,USAGE=A5    ,ACTUAL=A5       ,$
    FIELDNAME=OFF_ZIP_L ,ALIAS=      ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=O_PHONES    ,ALIAS=      ,USAGE=A21   ,ACTUAL=A21      ,$
  FIELDNAME=OFF_AREA_CDE,ALIAS=      ,USAGE=A3    ,ACTUAL=A3       ,$
  FIELDNAME=SPEED_DIAL  ,ALIAS=      ,USAGE=A3    ,ACTUAL=A3       ,$

SEGNAME=PHONES,PARENT=OFFICE,SEGTYPE=S,OCCURS=3,POSITION=O_PHONES ,$
  FIELDNAME=OFF_PHONE   ,ALIAS=      ,USAGE=A7    ,ACTUAL=A7       ,$
  FIELDNAME=LINE_NO     ,ALIAS=ORDER ,USAGE=I4    ,ACTUAL=I4       ,$

SEGNAME=STRUCTUR,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=STRUCTURE_CD,ALIAS=      ,USAGE=A2    ,ACTUAL=A2       ,$
  FIELDNAME=STRUCTURE_DT,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
```

```
SEGNAME=SUBORDS,PARENT=STRUCTUR,SEGTYPE=U,$
  FIELDNAME=SUB_ID      ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  GROUP=SUB_NAME        ,ALIAS=      ,USAGE=A25  ,ACTUAL=A25      ,$
    FIELDNAME=SUB_F_NAME,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
    FIELDNAME=SUB_L_NAME,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=SUB_STREET  ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20      ,$
  FIELDNAME=SUB_CITY    ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=SUB_STATE   ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  GROUP=SUB_FULL_ZIP    ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
    FIELDNAME=SUB_ZIP   ,ALIAS=      ,USAGE=A5   ,ACTUAL=A5       ,$
    FIELDNAME=SUB_ZIP_L ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=SUB_PHONE   ,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=SUB_STATUS  ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  FIELDNAME=SUB_SSN     ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
  FIELDNAME=SUB_STRT_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=SUB_TERM_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=SUB_BRTH_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$

SEGNAME=EMPOSIT,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=POS_STRT_DTE,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=POS_FIN_DTE ,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=SALARY_GRADE,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SALARY_AMT  ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=BONUS_PCT   ,ALIAS=      ,USAGE=P4   ,ACTUAL=P2       ,$
  FIELDNAME=COMMIS_PCT  ,ALIAS=      ,USAGE=P4   ,ACTUAL=P2       ,$
  FIELDNAME=OVERTIME_PCT,ALIAS=      ,USAGE=P5.2 ,ACTUAL=P2       ,$

SEGNAME=JOB,PARENT=EMPOSIT,SEGTYPE=U,$
  FIELDNAME=JOB_ID      ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=TITLE       ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20,
   FIELDTYPE=I,$
    DEFINE SHORTTITLE/A10 = EDIT(JTIT,'9999999999$');             ,$
  FIELDNAME=JOB_DESC    ,ALIAS=      ,USAGE=A120 ,ACTUAL=A120     ,$
  FIELDNAME=REQUIREMENTS,ALIAS=      ,USAGE=A120 ,ACTUAL=A120     ,$
  FIELDNAME=MIN_SALARY  ,ALIAS=      ,USAGE=P12.2,ACTUAL=Z8       ,$
  FIELDNAME=MAX_SALARY  ,ALIAS=      ,USAGE=P12.2,ACTUAL=Z8       ,$
  FIELDNAME=SAL_GRADE_1 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SAL_GRADE_2 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SAL_GRADE_3 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=SAL_GRADE_4 ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z2       ,$
  FIELDNAME=POSITION_NUM,ALIAS=      ,USAGE=P4   ,ACTUAL=Z3       ,$
  FIELDNAME=NUM_OPEN    ,ALIAS=      ,USAGE=P4   ,ACTUAL=Z3       ,$

SEGNAME=EXPERTSE,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=SKILL_LEVEL ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  FIELDNAME=EXPERT_DTE  ,ALIAS=      ,USAGE=A6YMD,ACTUAL=A6       ,$

SEGNAME=SKILL,PARENT=EXPERTSE,SEGTYPE=U,$
  FIELDNAME=SKILL_ID    ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=SKILL_NAME  ,ALIAS=      ,USAGE=A12  ,ACTUAL=A12,
   FIELDTYPE=I,$
  FIELDNAME=SKILL_DESC  ,ALIAS=      ,USAGE=A60  ,ACTUAL=A60      ,$
```

```
SEGNAME=COVERAGE,PARENT=EMPLOYE,SEGTYPE=S,$
  FIELDNAME=COV_SEL_DT   ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=COV_TERM_DTE,ALIAS=       ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=COVER_TYPE   ,ALIAS=      ,USAGE=A1   ,ACTUAL=A1       ,$
  FIELDNAME=COV_CODE     ,ALIAS=      ,USAGE=A3   ,ACTUAL=A3       ,$

SEGNAME=HOSPITAL,PARENT=COVERAGE,SEGTYPE=S,$
  FIELDNAME=H_CLAIM_DTE  ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=H_FIRST_NAME,ALIAS=       ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=H_LAST_NAME  ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=H_BIRTH_DTE  ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=H_SEX        ,ALIAS=      ,USAGE=A1   ,ACTUAL=A1       ,$
  FIELDNAME=H_RELATED_BY,ALIAS=       ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=HOSP_NAME    ,ALIAS=      ,USAGE=A25  ,ACTUAL=A25      ,$
  FIELDNAME=HOSP_STREET  ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20      ,$
  FIELDNAME=HOSP_CITY    ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=HOSP_STATE   ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  GROUP=HOSP_FUL_ZIP     ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
    FIELDNAME=HOSP_ZIP   ,ALIAS=      ,USAGE=A5   ,ACTUAL=A5       ,$
    FIELDNAME=HOSP_ZIP_L,ALIAS=       ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=ADMITTED     ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=DISCHARGED   ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=H_DIAGNOSIS1,ALIAS=       ,USAGE=A60  ,ACTUAL=A60      ,$
  FIELDNAME=H_DIAGNOSIS2,ALIAS=       ,USAGE=A60  ,ACTUAL=A60      ,$
  FIELDNAME=WARD_DAYS    ,ALIAS=      ,USAGE=P5   ,ACTUAL=P3       ,$
  FIELDNAME=WARD_RATE    ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=WARD_TOTAL   ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=SEMI_DAYS    ,ALIAS=      ,USAGE=P5   ,ACTUAL=P3       ,$
  FIELDNAME=SEMI_RATE    ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=SEMI_TOTAL   ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=DELIVERY_TOT,ALIAS=       ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=ANESTHES_TOT,ALIAS=       ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=LAB_TOT      ,ALIAS=      ,USAGE=P10.2,ACTUAL=P5       ,$
  FIELDNAME=             ,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=CLAIM_MONTH  ,ALIAS=CMO   ,USAGE=I2   ,ACTUAL=Z2       ,$
```

```
SEGNAME=NON_HOSP,SEGTYPE=S,PARENT=COVERAGE,$
  FIELDNAME=N_CLAIM_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=N_FIRST_NAME,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=N_LAST_NAME ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=N_BIRTH_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=N_SEX       ,ALIAS=      ,USAGE=A1   ,ACTUAL=A1       ,$
  FIELDNAME=N_RELATED_BY,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=PHYS_FNAME  ,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=PHYS_LNAME  ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=PHYS_STREET ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20      ,$
  FIELDNAME=PHYS_CITY   ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=PHYS_STATE  ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  GROUP=PHYS_FUL_ZIP    ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
    FIELDNAME=PHYS_ZIP  ,ALIAS=      ,USAGE=A5   ,ACTUAL=A5       ,$
    FIELDNAME=PHYS_ZIP_L,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=PHYS_ID     ,ALIAS=      ,USAGE=P6   ,ACTUAL=Z6       ,$
  FIELDNAME=P_DIAGNOSIS1,ALIAS=      ,USAGE=A60  ,ACTUAL=A60      ,$
  FIELDNAME=P_DIAGNOSIS2,ALIAS=      ,USAGE=A60  ,ACTUAL=A60      ,$
  FIELDNAME=P_NO_OF_PROC,ALIAS=      ,USAGE=I2   ,ACTUAL=I2       ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A1   ,ACTUAL=A1       ,$

SEGNAME=PHYSCHRG,SEGTYPE=S,PARENT=NON_HOSP,OCCURS=P_NO_OF_PROC    ,$
  FIELDNAME=P_SERVICE_DT,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=PHYS_PROC_CD,ALIAS=      ,USAGE=P4   ,ACTUAL=Z4       ,$
  FIELDNAME=P_SERV_DESC ,ALIAS=      ,USAGE=A60  ,ACTUAL=A60      ,$
  FIELDNAME=PHYS_FEE    ,ALIAS=      ,USAGE=P11.2,ACTUAL=P5       ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A1   ,ACTUAL=A1       ,$
  FIELDNAME=PHYS_CHRG_NO,ALIAS=ORDER ,USAGE=I4   ,ACTUAL=I4       ,$

SEGNAME=DENTAL,SEGTYPE=S,PARENT=COVERAGE,$
  FIELDNAME=D_CLAIM_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=D_FIRST_NAME,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=D_LAST_NAME ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=D_BIRTH_DTE ,ALIAS=      ,USAGE=I6YMD,ACTUAL=Z6       ,$
  FIELDNAME=D_SEX       ,ALIAS=      ,USAGE=A1   ,ACTUAL=A1       ,$
  FIELDNAME=D_RELATED_BY,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=DENT_FNAME  ,ALIAS=      ,USAGE=A10  ,ACTUAL=A10      ,$
  FIELDNAME=DENT_LNAME  ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=DENT_STREET ,ALIAS=      ,USAGE=A20  ,ACTUAL=A20      ,$
  FIELDNAME=DENT_CITY   ,ALIAS=      ,USAGE=A15  ,ACTUAL=A15      ,$
  FIELDNAME=DENT_STATE  ,ALIAS=      ,USAGE=A2   ,ACTUAL=A2       ,$
  GROUP=DENT_FUL_ZIP    ,ALIAS=      ,USAGE=A9   ,ACTUAL=A9       ,$
    FIELDNAME=DENT_ZIP  ,ALIAS=      ,USAGE=A5   ,ACTUAL=A5       ,$
    FIELDNAME=DENT_ZIP_L,ALIAS=      ,USAGE=A4   ,ACTUAL=A4       ,$
  FIELDNAME=DENT_LICENSE,ALIAS=      ,USAGE=P6   ,ACTUAL=Z6       ,$
  FIELDNAME=D_NO_OF_PROC,ALIAS=      ,USAGE=I2   ,ACTUAL=I2       ,$
  FIELDNAME=            ,ALIAS=      ,USAGE=A3   ,ACTUAL=A3       ,$
```

```
SEGNAME=DENTCHRG,SEGTYPE=S,PARENT=DENTAL,OCCURS=D_NO_OF_PROC,$
  FIELDNAME=TOOTH_NUM    ,ALIAS=       ,USAGE=P2    ,ACTUAL=Z2       ,$
  FIELDNAME=D_SERVICE_DT,ALIAS=       ,USAGE=A6YMD,ACTUAL=A6       ,$
  FIELDNAME=DENT_PROC_CD,ALIAS=       ,USAGE=P4    ,ACTUAL=Z4       ,$
  FIELDNAME=D_SERV_DESC  ,ALIAS=       ,USAGE=A60   ,ACTUAL=A60      ,$
  FIELDNAME=DENT_FEE     ,ALIAS=       ,USAGE=P11.2,ACTUAL=P5       ,$
  FIELDNAME=             ,ALIAS=       ,USAGE=A3    ,ACTUAL=A3       ,$
  FIELDNAME=DENT_CHRG_NO,ALIAS=ORDER ,USAGE=I9    ,ACTUAL=I4       ,$

SEGNAME=INSURNCE,PARENT=COVERAGE,SEGTYPE=U,$
  FIELDNAME=INS_PLAN_CDE,ALIAS=       ,USAGE=A3    ,ACTUAL=A3       ,$
  FIELDNAME=INS_CO_NAME  ,ALIAS=       ,USAGE=A45   ,ACTUAL=A45      ,$
  FIELDNAME=INS_STREET   ,ALIAS=       ,USAGE=A20   ,ACTUAL=A20      ,$
  FIELDNAME=INS_CITY     ,ALIAS=       ,USAGE=A15   ,ACTUAL=A15      ,$
  FIELDNAME=INS_STATE    ,ALIAS=       ,USAGE=A2    ,ACTUAL=A2       ,$
  GROUP=INS_FULL_ZIP     ,ALIAS=       ,USAGE=A9    ,ACTUAL=A9       ,$
    FIELDNAME=INS_ZIP    ,ALIAS=       ,USAGE=A5    ,ACTUAL=A5       ,$
    FIELDNAME=INS_ZIP_L  ,ALIAS=       ,USAGE=A4    ,ACTUAL=A4       ,$
  FIELDNAME=INS_PHONE    ,ALIAS=       ,USAGE=A10   ,ACTUAL=A10      ,$
  FIELDNAME=INS_GROUP_NO,ALIAS=       ,USAGE=A6    ,ACTUAL=A6       ,$
  FIELDNAME=DEDUCT       ,ALIAS=       ,USAGE=P12.2,ACTUAL=P5       ,$
  FIELDNAME=MAX_LIFE_CST,ALIAS=       ,USAGE=P12.2,ACTUAL=P5       ,$
  FIELDNAME=FAMILY_COST  ,ALIAS=       ,USAGE=P12.2,ACTUAL=P5       ,$
  FIELDNAME=DEPENDNT_CST,ALIAS=       ,USAGE=P12.2,ACTUAL=P5       ,$
```

Access File for Network

This Access File is associated with the network subschema EMPSS01 and corresponds to its Master File:

```
SSCHEMA=EMPSS01,RELEASE=15,MODE=DML,TRACE=NO,READY=,$
SEGNAM=DEPT,RECORD=DEPARTMENT,AREA=ORG-DEMO-REGION,
  CLCFLD=DEPT_ID,CLCDUP=N,$

SEGNAM=EMPLOYE,RECORD=EMPLOYEE,AREA=EMP-DEMO-REGION,
  CLCFLD=EMP_ID,CLCDUP=N,ACCESS=SET,SETNAME=DEPT-EMPLOYEE,
  SETMBR=OA,GETOWN=Y,MULTMBR=N,$

SEGNAM=OFFICE,RECORD=OFFICE,AREA=ORG-DEMO-REGION,
  CLCFLD=OFF_CODE,CLCDUP=N,ACCESS=SET,SETNAME=OFFICE-EMPLOYEE,
  SETMBR=OA,GETOWN=Y,MULTMBR=N,$

SEGNAM=STRUCTUR,RECORD=STRUCTURE,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=MANAGES,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=SUBORDS,RECORD=EMPLOYEE,AREA=EMP-DEMO-REGION,
  CLCFLD=SUB_ID,CLCDUP=N,ACCESS=SET,SETNAME=REPORTS-TO,
  SETMBR=OM,GETOWN=Y,MULTMBR=N,$

SEGNAM=EMPOSIT,RECORD=EMPOSITION,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-EMPOSITION,SETMBR=MA,GETOWN=Y,MULTMBR=N,$
```

```
SEGNAM=JOB,RECORD=JOB,AREA=ORG-DEMO-REGION,
  CLCFLD=JOB_ID,CLCDUP=N,ACCESS=SET,SETNAME=JOB-EMPOSITION,
  SETMBR=OM,GETOWN=Y,MULTMBR=N,SEQFIELD=TITLE,$

SEGNAM=EXPERTSE,RECORD=EXPERTISE,AREA=EMP-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-EXPERTISE,KEYFLD=SKILL_LEVEL,SETORD=D,
  SETDUP=Y,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=SKILL,RECORD=SKILL,AREA=ORG-DEMO-REGION,
  CLCFLD=SKILL_ID,CLCDUP=N,ACCESS=SET,SETNAME=SKILL-EXPERTISE,
  KEYFLD=SKILL_LEVEL,SETORD=D,SETDUP=Y,
  SETMBR=MA,GETOWN=Y,MULTMBR=N,SEQFIELD=SKILL_NAME,$

SEGNAM=COVERAGE,RECORD=COVERAGE,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=EMP-COVERAGE,SETMBR=MA,GETOWN=Y,MULTMBR=N,$

SEGNAM=HOSPITAL,RECORD=HOSPITAL-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

SEGNAM=NON_HOSP,RECORD=NON-HOSP-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

SEGNAM=DENTAL,RECORD=DENTAL-CLAIM,AREA=INS-DEMO-REGION,
  ACCESS=SET,SETNAME=COVERAGE-CLAIMS,SETMBR=MA,GETOWN=Y,MULTMBR=Y,$

IXSET=JOB-TITLE-NDX,IXFLD=TITLE,IXDUP=N,IXORD=A,
  IXAREA=INS-DEMO-REGION,$

IXSET=SKILL-NAME-NDX,IXFLD=SKILL_NAME,IXDUP=N,IXORD=D,
  IXAREA=EMP-DEMO-REGION,$

SEGNAM=INSURNCE,RECORD=INSURANCE-PLAN,AREA=INS-DEMO-REGION,
  CLCFLD=INS_PLAN_CDE,CLCDUP=N,ACCESS=CLC,KEYFLD=COV_CODE,$
```

LRF Subschema: EMPSS02

This subschema shows the LRF view of the schema EMPSCHM. It contains the following items:

- Physical record types that are used to create logical records.
- A SELECT clause for CALC access.
- A SELECT ELEMENT clause.
- A SELECT NULL clause.
- A SELECT INDEX clause.

```
ADD SUBSCHEMA NAME IS EMPSS02
    OF SCHEMA NAME IS EMPSCHM VERSION  1
    USAGE IS LR
DMCL NAME IS EMPDMCL
    OF SCHEMA NAME IS EMPSCHM VERSION  1
COMMENTS 'THIS IS THE COMPLETE VIEW OF EMPSCHM'.
ADD AREA NAME IS EMP-DEMO-REGION.
ADD AREA NAME IS ORG-DEMO-REGION.
ADD RECORD NAME IS DEPARTMENT.
ADD RECORD NAME IS EMPLOYEE.
ADD RECORD NAME IS EMPOSITION.
ADD RECORD NAME IS JOB.
ADD SET DEPT-EMPLOYEE.
ADD SET EMP-NAME-NDX.
ADD SET EMP-EMPOSITION.
ADD SET JOB-EMPOSITION.
ADD SET JOB-TITLE-NDX.

ADD
   LOGICAL RECORD NAME IS DEPT-EMP-POS
   ELEMENTS ARE DEPARTMENT
                EMPLOYEE
                EMPOSITION.

ADD
PATH-GROUP NAME IS OBTAIN DEPT-EMP-POS
   SELECT FOR FIELDNAME-EQ DEPT-ID-0410
      OBTAIN DEPARTMENT
        WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION

   SELECT FOR FIELDNAME-EQ EMP-ID-0415
      OBTAIN EMPLOYEE
        WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      IF DEPT-EMPLOYEE MEMBER
      OBTAIN OWNER WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION

   SELECT FOR ELEMENT DEPARTMENT
      OBTAIN EACH DEPARTMENT WITHIN ORG-DEMO-REGION
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
```

```
   SELECT FOR ELEMENT EMPLOYEE
      OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
      IF DEPT-EMPLOYEE MEMBER
      OBTAIN OWNER WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION


   SELECT FOR ELEMENT EMPOSITION
      OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
      OBTAIN OWNER WITHIN EMP-EMPOSITION
      IF DEPT-EMPLOYEE MEMBER
      OBTAIN OWNER WITHIN DEPT-EMPLOYEE


   SELECT
      OBTAIN EACH DEPARTMENT WITHIN ORG-DEMO-REGION
      IF DEPT-EMPLOYEE IS NOT EMPTY
      OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      IF EMP-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION.


ADD
   LOGICAL RECORD NAME IS JOB-EMPOSITION
   ELEMENTS ARE JOB
                EMPOSITION.


ADD
PATH-GROUP NAME IS OBTAIN JOB-EMPOSITION
   SELECT FOR FIELDNAME-EQ JOB-ID-0440
      OBTAIN JOB
        WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST
      IF JOB-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION


   SELECT USING INDEX JOB-TITLE-NDX
       FOR FIELDNAME TITLE-0440
      OBTAIN EACH JOB USING INDEX
      IF JOB-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION


   SELECT FOR FIELDNAME START-DATE-0420
      OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
      IF JOB-EMPOSITION MEMBER
      OBTAIN OWNER WITHIN JOB-EMPOSITION


   SELECT FOR ELEMENT JOB
      OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
      IF JOB-EMPOSITION IS NOT EMPTY
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
```

```
   SELECT FOR ELEMENT EMPOSITION
      OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
      IF JOB-EMPOSITION MEMBER
      OBTAIN OWNER WITHIN JOB-EMPOSITION

   SELECT
      OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
        ON 0307 CLEAR RETURN LR-NOT-FOUND
        ON 0000 NEXT
      IF JOB-EMPOSITION IS NOT EMPTY
        ON 0000 ITERATE
        ON 1601 NEXT
      OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION
        ON 0000 NEXT
        ON 0307 ITERATE.
GENERATE.
```

Master File for LRF

```
FILE=EMPDATA,SUFFIX=IDMSR,$
SEGNAME=DEPTEMPO,$
  FIELD=DEPT_ID     ,ALIAS=         ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=DEPT_NAME   ,ALIAS=         ,USAGE=A45   ,ACTUAL=A45   ,$
  FIELD=DEPT_HEAD   ,ALIAS=         ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=            ,ALIAS=FILL.END,USAGE=A3    ,ACTUAL=A3    ,$
  FIELD=EMP_ID      ,ALIAS=         ,USAGE=A4    ,ACTUAL=A4    ,$
  GROUP=EMP_NAME    ,ALIAS=         ,USAGE=A25   ,ACTUAL=A25   ,$
    FIELD=FIRST_NAME,ALIAS=         ,USAGE=A10   ,ACTUAL=A10   ,$
    FIELD=LAST_NAME ,ALIAS=         ,USAGE=A15   ,ACTUAL=A15   ,$
  FIELD=EMP_STREET  ,ALIAS=         ,USAGE=A20   ,ACTUAL=A20   ,$
  FIELD=EMP_CITY    ,ALIAS=         ,USAGE=A15   ,ACTUAL=A15   ,$
  FIELD=EMP_STATE   ,ALIAS=         ,USAGE=A2    ,ACTUAL=A2    ,$
  GROUP=EMP_FULL_ZIP,ALIAS=         ,USAGE=A9    ,ACTUAL=A9    ,$
    FIELD=EMP_ZIP   ,ALIAS=         ,USAGE=A5    ,ACTUAL=A5    ,$
    FIELD=EMP_ZIP_L ,ALIAS=         ,USAGE=A4    ,ACTUAL=A4    ,$
  FIELD=EMP_PHONE   ,ALIAS=         ,USAGE=A10   ,ACTUAL=A10   ,$
  FIELD=STATUS      ,ALIAS=         ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=SOC_SEC_NUM ,ALIAS=         ,USAGE=A9    ,ACTUAL=A9    ,$
  FIELD=EMP_STRT_DTE,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=EMP_TERM_DTE,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=EMP_BRTH_DTE,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=            ,ALIAS=FILL.END,USAGE=A6    ,ACTUAL=A6    ,$
  FIELD=POS_STRT_DT1,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=POS_FIN_DT1 ,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6    ,$
  FIELD=SALARY_GRAD1,ALIAS=         ,USAGE=P4    ,ACTUAL=Z2    ,$
  FIELD=SALARY_AMT1 ,ALIAS=         ,USAGE=P10.2,ACTUAL=P5    ,$
  FIELD=BONUS_PCT1  ,ALIAS=         ,USAGE=P4    ,ACTUAL=P2    ,$
  FIELD=COMMIS_PCT1 ,ALIAS=         ,USAGE=P4    ,ACTUAL=P2    ,$
  FIELD=OVERTIM_PCT1,ALIAS=         ,USAGE=P5.2  ,ACTUAL=P2    ,$
```

```
SEGNAME=JOBPOS,PARENT=DEPTEMPO,SEGTYPE=U,$
  FIELD=JOB_ID      ,ALIAS=         ,USAGE=A4    ,ACTUAL=A4   ,$
  FIELD=TITLE       ,ALIAS=         ,USAGE=A20   ,ACTUAL=A20  ,$
  FIELD=JOB_DESC    ,ALIAS=         ,USAGE=A120  ,ACTUAL=A120 ,$
  FIELD=REQUIREMENTS,ALIAS=         ,USAGE=A120  ,ACTUAL=A120 ,$
  FIELD=MIN_SALARY  ,ALIAS=         ,USAGE=P12.2,ACTUAL=Z8   ,$
  FIELD=MAX_SALARY  ,ALIAS=         ,USAGE=P12.2,ACTUAL=Z8   ,$
  FIELD=SAL_GRADE_1 ,ALIAS=         ,USAGE=P4    ,ACTUAL=Z2   ,$
  FIELD=SAL_GRADE_2 ,ALIAS=         ,USAGE=P4    ,ACTUAL=Z2   ,$
  FIELD=SAL_GRADE_3 ,ALIAS=         ,USAGE=P4    ,ACTUAL=Z2   ,$
  FIELD=SAL_GRADE_4 ,ALIAS=         ,USAGE=P4    ,ACTUAL=Z2   ,$
  FIELD=POSITION_NUM,ALIAS=         ,USAGE=P4    ,ACTUAL=Z3   ,$
  FIELD=NUM_OPEN    ,ALIAS=         ,USAGE=P4    ,ACTUAL=Z3   ,$
  FIELD=            ,ALIAS=FILL.END,USAGE=A2    ,ACTUAL=A2   ,$
  FIELD=POS_STRT_DT2,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6   ,$
  FIELD=POS_FIN_DT2 ,ALIAS=         ,USAGE=A6YMD,ACTUAL=A6   ,$
  FIELD=SALARY_GRAD2,ALIAS=         ,USAGE=P4    ,ACTUAL=Z2   ,$
  FIELD=SALARY_AMT2 ,ALIAS=         ,USAGE=P10.2,ACTUAL=P5   ,$
  FIELD=BONUS_PCT2  ,ALIAS=         ,USAGE=P4    ,ACTUAL=P2   ,$
  FIELD=COMMIS_PCT2 ,ALIAS=         ,USAGE=P4    ,ACTUAL=P2   ,$
  FIELD=OVERTIM_PCT2,ALIAS=         ,USAGE=P5.2  ,ACTUAL=P2   ,$
```

Access File for LRF

This Access File is associated with LRF subschema EMPSS02, and corresponds to its Master File.

```
SSCHEMA=EMPSS02,RELEASE=15,MODE=LR,TRACE=PARMS,READY=ALL,$
SEGNAM=DEPTEMPO,RECORD=DEPT-EMP-POS,AREA=EMP-DEMO-REGION,LR=Y,$
SEGNAM=JOBPOS,RECORD=JOB-EMPOSITION,AREA=ORG-DEMO-REGION,LR=Y,
 ACCESS=LR,KEYFLD=POS_STRT_DT1,IXFLD=POS_STRT_DT2,$
```

Sample of a Partial LRF Record

The following is an example of a NULL SELECT clause that creates a partial record by returning a user-defined record code. The adapter does not support this user-defined code or any status code other than LR-FOUND or LR-NOT-FOUND.

```
SELECT
   OBTAIN EACH JOB WITHIN ORG-DEMO-REGION
   IF JOB-EMPOSITION IS NOT EMPTY
     ON 0000 RETURN NO-POS-FOR-JOB
   OBTAIN EACH EMPOSITION WITHIN JOB-EMPOSITION.
```

SPF Indexes

The following is a section of a subschema that contains SPF indexes. Comparable Integrated Indexes are found in the LRF subschema EMPSS02 listed as EMP-NAME-NDX, JOB-TITLE-NDX, and SKILL-NAME-NDX.

```
ADD SET NAME IS IX-EMP-LNAME
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
     NEXT DBKEY POSITION IS AUTO
  MEMBER IS EMPLOYEE
     NEXT DBKEY POSITION IS AUTO
     OPTIONAL MANUAL
     ASCENDING KEY IS ( EMP-LAST-NAME-0415)
          DUPLICATES LAST
.

ADD SET NAME IS IX-TITLE
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
     NEXT DBKEY POSITION IS AUTO
  MEMBER IS JOB
     NEXT DBKEY POSITION IS AUTO
     OPTIONAL MANUAL
     DESCENDING KEY IS ( TITLE-0440 )
          DUPLICATES NOT ALLOWED
.

ADD SET NAME IS IX-SKILL-NAME
  ORDER IS SORTED
  MODE IS CHAIN
  OWNER IS IXOWNER
     NEXT DBKEY POSITION IS AUTO
  MEMBER IS SKILL
     NEXT DBKEY POSITION IS AUTO
     OPTIONAL MANUAL
     ASCENDING KEY IS ( SKILL-NAME-0455 )
          DUPLICATES NOT ALLOWED
    .
```

File Retrieval

The Adapter for IDMS/DB retrieves the necessary records that fulfill a request. It uses information from the following sources to choose the most appropriate and efficient retrieval method:

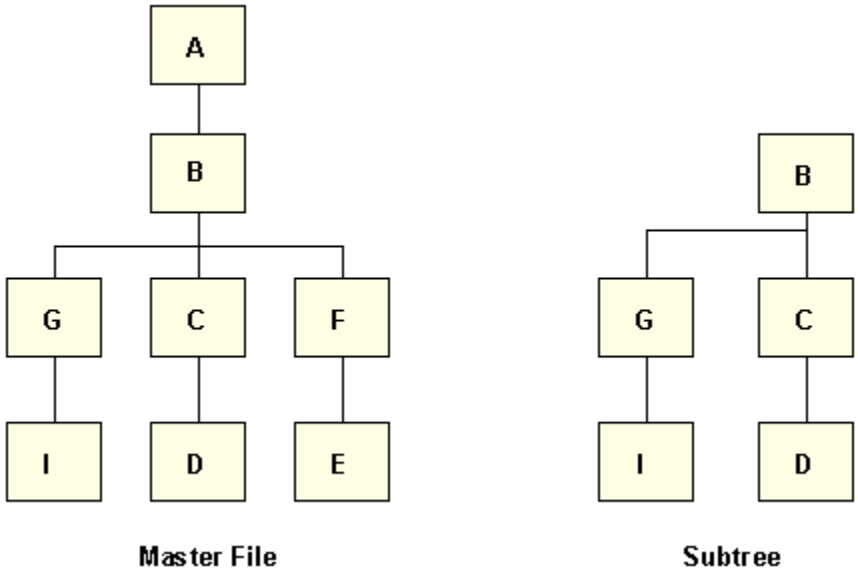
The server uses the Master File to select segments and retrieve data from them.

Note: Sections pertaining to unique segments discuss rules that apply regardless of whether the unique segment is the owner of its parent or is related to its parent through a CALC field, index, or LRF field.

Retrieval Subtree

To retrieve records for a request, the server first constructs a smaller subtree structure from the structure defined by the Master File. The subtree consists of segments that contain fields named explicitly in the request and those named implicitly by DBA DEFINE or COMPUTE statements. The subtree also includes any segments needed to connect these segments.

For example, if an SQL request needs fields from segments D, G, and I, the server constructs the subtree diagram shown below. Segments C and B do not contain fields needed for the request, but they are included in the subtree to connect segment I with segments D and G. Since the segments are descendants of segment B, the entry or root segment of the subtree is segment B. The Master File root segment A is not included, because its fields are not referenced; the records corresponding to segment B can be obtained independently of A. However, if segment B were an OCCURS segment, the IDMS/DB calls would be issued for segment A to obtain information for B.

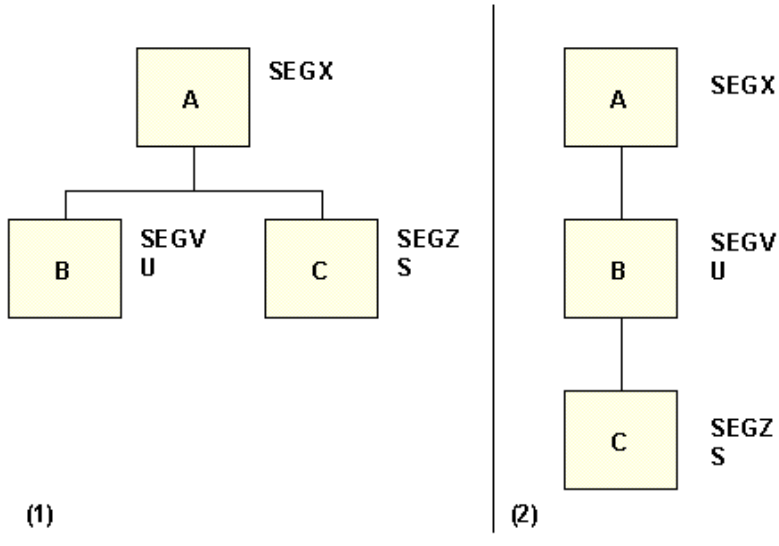


Retrieval Sequence With Unique Segments

The retrieval sequence for subtrees containing unique segments is still top-to-bottom, left-to-right, but the unique segments are treated as extensions of their parents. Records in a unique segment correspond one-to-one with the records in a parent. Records in a non-unique segment have a one-to-many correspondence. In cases where the parent segment has unique and non-unique descendants, the unique descendants are always retrieved first regardless of the left-to-right order.

A retrieval view shows which sort and WHERE clauses are valid. For sort statements (BY or ACROSS), the segment containing the sortfield must lie on the same path as the segments with all requested fields. That is, the segment with the BY or ACROSS field must be an ancestor or descendant of the segments containing the required fields.

For instance, panel 1 of the following diagram shows two descendant segments. The statement SELECT B ORDER BY C is invalid, because the segments containing fields B and C do not lie on the same path. However, if the segment containing B is a unique segment, the two segments do lie on the same path. Panel 2 shows the retrieval view. The statement SELECT B ORDER BY C is valid.



The retrieval sequence for unique segments may also affect the results of an SQL statement that contains COUNT or SUM. If a segment is the parent of a unique descendant, there is a one-to-one relationship. A COUNT statement, such as COUNT A AND B, returns identical results for each field, because the same record A is counted several times for each record B. If the parent/descendant relationship is reversed with a non-unique parent, the result for field A is a greater number than the result for field B.

Screening Conditions

If a record in a segment fails a WHERE condition, the server does not retrieve the corresponding records in descendant segments. Suppose this request is entered against the structure EMPSS01 (corresponding Master File is EMPFULL).

```
SELECT EMP_ID,OFF_CODE
FROM EMPFULL
WHERE DEPT_NAME = 'PERSONNEL'
ORDER BY OFF_CODE
```

Every time a record in segment DEPT has a value in the DEPT_NAME field not equal to PERSONNEL, the server ignores the corresponding records in descendant segments EMPLOYEE and OFFICE, and retrieves the next record in segment DEPT. In addition, when a WHERE clause on a lower segment fails, the row is removed from the server answer set.

To increase I/O efficiency, place the WHERE clauses at a higher level in the file structure. This restricts the number of records the server has to test. The example below shows the benefits of two WHERE clauses versus one.

Assume a subtree has four segments:

- DEPT contains department IDs and information.
- EMPLOYEE contains employee names and IDs.
- EMPOSIT contains the positions that the employee has held.
- JOB contains a list of jobs offered by the company.

To list all employees who are programmer/analysts:

```
SELECT TITLE,DEPT,LAST_NAME,FIRST_NAME
FROM EMPFULL
WHERE TITLE = 'PROGRAMMER/ANALYST'
ORDER BY DEPT,LAST_NAME,FIRST_NAME
```

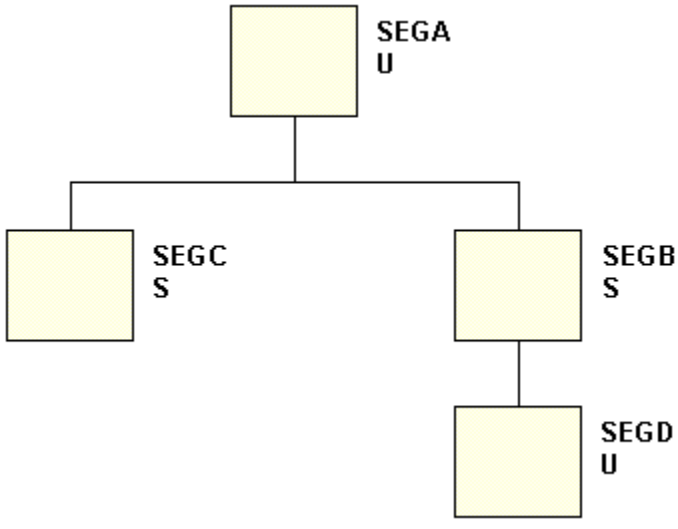
For this request with only one WHERE clause, the server retrieves each DEPT record, each EMPLOYEE record for a given DEPT, each EMPOSIT record for a given EMPLOYEE record, and the JOB record connected to each EMPOSIT. After retrieval, the server determines whether to include in the answer set the record from the value of the TITLE field. To make retrieval more efficient, add another WHERE clause on a segment higher in the structure. In this company, only the Internal Software department has programmer/analysts working for it:

```
SELECT TITLE,DEPT,LAST_NAME,FIRST_NAME
FROM EMPFULL
WHERE DEPT_NAME = 'INTERNAL SOFTWARE'
WHERE TITLE = 'PROGRAMMER/ANALYST'
```

Now the server retrieves and tests records only when the DEPT_NAME field equals the value INTERNAL SOFTWARE.

Screening Conditions With Unique Segments

If a record in a unique segment fails a WHERE test, the server rejects its parent and retrieves the next record of the parent segment. For example, in the following diagram, if a record in non-unique segment C fails a WHERE clause, the server retrieves the next record in segment C. Only if all C records for a given A fail the test is the A record rejected. When a record in unique segment D fails a test, the server rejects the parent B record and retrieves the next record in segment B. When a record in the entry A segment fails a test, the server retrieves the next A record, even if the entry segment is defined as unique.



Short Paths

When the server retrieves a record in a parent segment, it retrieves the corresponding records in the descendant segment. If descendant records do not exist, the processing of the parent record and whether it is included in an answer set depends on whether the descendant segment is unique or non-unique.

Short Paths in Unique Descendants

For a unique descendant with a missing record, the server creates a temporary record to replace the missing record. The temporary record contains fields with default values: blanks for alphanumeric fields and zeroes for numeric fields.

For example, an EMPLOYEE segment with the field EMP_NAME has a unique descendant OFFICE segment with the field OFF_CITY. The field OFF_CITY indicates the location of an employee office. Gary Smith does not work out of an office location, so he has no OFFICE record. In this situation, all requests that refer to OFF_CITY display blank spaces for the entry GARY SMITH.

Short Paths in Non-Unique Descendants

For a non-unique descendant segment with a missing record, the server rejects the parent instance and retrieves the next parent instance.

Syntax: How to Specify Short Paths in Non-Unique Descendants

For a non-unique descendant segment with a missing record, the results depend on how the ALL parameter is set:

```
SET ALL = {ON|OFF}
```

where:

`ON`

The parent record is processed provided that there are no screening conditions on fields in the descendant segment. Missing data is usually indicated on the report by the default NODATA character (.).

`OFF`

The parent instance is rejected and the next parent instance is retrieve. OFF is the default value.

Note: SET ALL = PASS is *not* supported by the adapter.

Record Retrieval

To obtain all of the necessary records to fulfill a request, the Adapter for IDMS/DB navigates the IDMS/DB database using DML or LRF commands. The adapter automatically generates DML or LRF commands based on information from the Master File, Access File, and your request for the most appropriate and efficient IDMS/DB retrieval method.

There are three kinds of IDMS/DB access:

- *Entry Segment Retrieval of Network Records*.
- *Descendant Segment Retrieval of Network Records*.
- *LRF Record Retrieval* (LR and ASF).

Subsequent sections discuss the navigational strategies used for each kind of access.

Entry Segment Retrieval of Network Records

The server constructs a retrieval subtree based on the Master and Access Files and your request. The root of this subtree is called the entry segment, because the server begins its retrieval search of the database at that point. The actual IDMS/DB retrieval calls used on the entry segment depend on the entry segment's Access File information and any WHERE clauses. To perform the most efficient record retrieval on the entry segment, the adapter chooses one of the following techniques:

- *Retrieval by Database Key*
- *Retrieval by CALC Field*
- *Retrieval by Index*
- *SEQFIELD Parameter*
- *Retrieval by Area Sweep*

These techniques are listed in descending order of efficiency. The idea behind selection logic is to perform as many WHERE clauses as possible at the IDMS/DB level. This minimizes the actual I/O operations required to access the necessary data. Area sweeps are the least desirable retrieval technique, because they read through every record type in the named area, including record types that correspond to other segments, and return every entry segment record to the server. At this point, the server selects those records that satisfy the request's test criteria and discards the rest.

Retrieval by Database Key

The IDMS/DB database key method of retrieval takes precedence over the other methods because it is the most efficient. This method depends on the existence of two conditions:

- A field that corresponds to the IDMS/DB database key (ALIAS=DBKEY) for the entry segment.
- An equality test in the request on the DBKEY field.

The equality test sets the field name of the DBKEY from the entry segment in the Master File equal to a specified numeric value(s):

```
WHERE field = 'value1'
```

Retrieval by CALC Field

If there is no WHERE clause on the DBKEY for the entry segment, the second choice is CALC access. Retrieval through the CALC key, while not as efficient as DBKEY access, takes precedence over an index or area sweep retrieval.

The CALC retrieval method depends on the existence of two conditions:

- The entry segment must contain a CALC keyfield as specified in the Access File.
- A fully qualified equality test in the request on the CALC field.

The WHERE clause sets the field name of the CALC key for the segment equal to fully qualified values:

```
WHERE field = 'value1'
```

For each value specified in the WHERE clause, the adapter calls IDMS/DB with the following DML command:

```
OBTAIN CALC record
```

If the Access File indicates duplicate records (CLCDUP=Y), each DML call is followed by subsequent calls:

```
OBTAIN DUPLICATE record
```

This ensures that all appropriate records are obtained to satisfy the request.

Retrieval by Index

If there is no DBKEY or CALC key test criteria in the request, the adapter selects the index retrieval method.

Note: An index field must be defined with the FIELDTYPE=I attribute in the Master File and a corresponding index declaration must exist in the Access File.

Index retrieval is performed using:

- WHERE clauses for an indexed field or GROUP in the entry segment.
- The optional parameter SEQFIELD in the Access File segment declaration.

The first method requires at least one WHERE clause that specifies the field name of the index field. The following WHERE clause invokes index-based retrieval:

- Fully qualified.

  ```
  WHERE = 'PROGRAMMER/ANALYST'
  ```

- Partially qualified (generic), also called masking. This applies to alphanumeric fields only.

```
WHERE TITLE LIKE 'PROGRAMMER%'
```

- Specified as a range of values.

```
WHERE TITLE BETWEEN 'PROGRAMMER' AND 'WORD PROCESSOR'
```

When two or more WHERE clauses in a request qualify an index on the entry-level segment, fully qualified retrieval takes precedence over generic; generic retrieval takes precedence over range. If two WHERE clauses are the same type, the index in the first WHERE clause is used. All types of WHERE clauses are also supported for indices that allow duplicate values.

If the test criteria indicates index retrieval, the adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN setname USING value
```

In this command, *setname* is the name of the index set specified in the Access File.

Then, if the Access File indicates duplicate records, the adapter issues this DML command for index sets:

```
OBTAIN NEXT record WITHIN setname
```

The above OBTAIN NEXT call is issued until all the duplicate records are retrieved. This same NEXT call is also issued if your request contains generic or range WHERE clauses. It is performed once for every value or range specified in the WHERE clause.

Note: If the IXORD parameter is improperly specified in the Access File, a range WHERE clause may erroneously produce an answer set with one or no records.

SEQFIELD Parameter

The second method of index retrieval does not require WHERE clauses, and yet prohibits area sweeps on entry segments. To use this method, add the optional SEQFIELD parameter to the Access File and specify the name of the indexed field as the value of SEQFIELD.

When your request does not contain a WHERE (for DBKEY, CALC key, or another index) and a SEQFIELD is specified for the entry segment, the adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN setname
```

In this command, *setname* is the IDMS/DB set name of the indexed field (IXSET parameter) from the Access File.

The adapter then issues the next command until all records connected to the index set are retrieved:

```
OBTAIN NEXT record WITHIN setname
```

If no sort criteria (ORDER BY) is specified in the request, the answer set is produced in ascending or descending index set order.

The SEQFIELD method is recommended for indexed segments in large IDMS/DB databases where only a small percentage of record occurrences in a given area are the record types defined by the segments. In such cases, IDMS/DB resource utilization can be greatly reduced through the use of this parameter.

Note: If the index set connection is not mandatory/automatic (MA), some of the records in this record type may not be accessed if it is the entry segment. In this situation, only records that are members of the index set are supplied to the server. If this retrieval result is undesirable, you should omit the SEQFIELD parameter.

Retrieval by Area Sweep

An area sweep is the least efficient method of entry-level retrieval, because it reads through every record in an IDMS/DB area to return records of a given record type. Despite its inefficiency, an area sweep is sometimes the only method available for retrieval.

The adapter performs an area sweep if one of the following occurs:

- No equality WHERE on the DBKEY for the root exists.
- No equality WHERE on the CALC key for the root exists.
- No equality or range WHERE on an indexed field exists.
- No SEQFIELD parameter is specified.

If one of the above situations occurs, the adapter issues this DML command to IDMS/DB:

```
OBTAIN FIRST record WITHIN areaname
```

In this command, *areaname* is the name of the IDMS/DB database area specified in the Access File. Next, the adapter continues to issue this command until all the records are obtained:

```
OBTAIN NEXT record WITHIN areaname
```

Descendant Segment Retrieval of Network Records

To retrieve records from a descendant segment, the adapter's navigational strategy depends on Access File parameters and the SEGTYPE parameter in the Master File. In a few cases, the WHERE clauses in a request also affect the strategy.

In general, the ACCESS parameter in the Access File determines retrieval strategy, because it indicates how parent/descendant relationships are implemented. There is a retrieval strategy for each kind of relationship:

- *Set-Based Retrieval*
- *CALC-Based Retrieval*
- *Index-Based Retrieval*
- *LRF Record Retrieval*
- *Overriding DBNAME and DICTNAME*

Set-Based Retrieval

For a set relationship (ACCESS=SET), the IDMS/DB set is searched starting with the owner to obtain related member record(s). Set relationships are physical ones, implemented by set pointer chains. The SEGTYPE parameter in the Master File indicates whether the descendant segment is unique or non-unique.

If the descendant segment is non-unique (SEGTYPE=S), it represents a member record type. For non-unique descendants, the adapter issues this command:

```
OBTAIN NEXT record WITHIN setname
```

This command is repeated until IDMS/DB indicates that the end of the set is reached. Then the adapter obtains records of other descendant segments for the same parent segment. If no other descendant segments exist, the next parent record is retrieved.

If the descendant segment is unique (SEGTYPE=U), it represents an owner record type. For unique descendant segments, the adapter issues this command:

```
OBTAIN OWNER WITHIN setname
```

The command is issued once, since there is one owner per set. The adapter continues to retrieve descendant records for the same parent or retrieves the next parent record.

The KEYFLD and MULTMBR parameters in the Access File also affect retrieval for certain requests. For a sorted set, the KEYFLD parameter specifies that the set is ordered by a specified field. When the request references a field from the parent segment and has a WHERE (=, BETWEEN, >, >=, <, <=) on the sortfield, the adapter sends this command to IDMS/DB:

```
OBTAIN record WITHIN setname USING value
```

If there are duplicate records (SETDUP=Y), the adapter issues this command:

```
OBTAIN NEXT record WITHIN setname
```

When the value of the sortfield changes beyond the specified range, retrieval for that segment stops. I/O operations are minimized when the sortfield value is supplied in the OBTAIN command.

Note: The means of implementing the sorted set (the traditional method or using IDMS/DB Integrated Indices) is transparent to the adapter.

The MULTMBR parameter indicates an IDMS/DB multi-member set. When it is specified, the adapter searches for other member record types (segments) in the Access File with the same setname. As a result, all necessary IDMS/DB areas are activated.

CALC-Based Retrieval

CALC-based relationships (ACCESS=CLC) are performed with embedded cross-references. A field in the parent segment corresponds to the CALC field in its descendant segment. The adapter uses the value from the parent field and performs entry-level IDMS/DB retrieval. The process of retrieving records from a descendant segment is similar to that of an entry segment. The difference is that the value supplied by the parent segment acts as the WHERE clause as if it were an explicit WHERE.

After the adapter retrieves the host field value (KEYFLD=value) from the parent segment, it calls IDMS/DB:

```
OBTAIN CALC record
```

Then, if the descendant segment is non-unique (SEGTYPE=S), the adapter issues this command until all of the appropriate records are obtained to satisfy the request:

```
OBTAIN DUPLICATE record
```

Note: If the CLCDUP parameter does not correspond to the SEGTYPE parameter, message EDA919 displays.

A descendant segment with a CALC-based relationship (ACCESS=CLC) may act as a parent and be related to its descendants using set-, CALC-, or index-based relationships.

Index-Based Retrieval

Like CALC-based relationships, index-based relationships (ACCESS=IX) also use embedded cross-references. In index-based relationships, the field in a descendant segment represents an index on the IDMS/DB record type. The index can be either a Sequential Processing Facility (SPF) index or an Integrated Index. The adapter uses the value from the parent segment and performs entry-level IDMS/DB retrieval by searching the index set. The process of retrieving records from a descendant segment is similar to that of an entry segment. The difference is that the value supplied by the parent segment acts as the WHERE clause, as if it were an explicit WHERE.

After the adapter retrieves the host field value (KEYFLD=value) from the parent segment, it calls IDMS/DB:

```
OBTAIN FIRST record WITHIN setname USING value
```

Then, if the descendant segment is non-unique (SEGTYPE=S), the adapter issues the following command until all indexed records with the host value are retrieved:

```
OBTAIN NEXT record WITHIN setname
```

Note: If the IXDUP parameter does not correspond to the SEGTYPE parameter, message EDA919 displays.

Only a descendant segment with an Integrated Index may act as a parent and be related to its descendants using set-, CALC-, or index-based relationships.

LRF Record Retrieval

To retrieve LR and ASF records, the adapter sends LRF calls to an access module IDMS/DB which invokes the Logical Record Facility program.

LRF-based records are retrieved when the Access File specifies MODE=LR in the subschema declaration and the adapter constructs an LR call to IDMS/DB with explicit or implicit WHERE clauses from the request. The LRF processes the request as generated by the adapter, selects the appropriate LR path, and constructs each flat view using the full set of WHERE clauses. The process is highly efficient in terms of I/O; only those records which pass the WHERE clause are passed back to the adapter from IDMS/DB.

The retrieval process for LRF records is identical to that of network record types, but the Logical Record Facility maintains its own navigational information for the database, selects the retrieval strategy most appropriate for a given request, and maintains its own set of occurrences.

When the subschema mode is Logical Record (MODE=LR), the adapter analyzes the request and creates an LRF command to be sent to the Logical Record Facility. The LRF command has two formats. The first format is for an entry segment without WHERE clauses:

```
OBTAIN NEXT record
```

The second format is for an entry with WHERE clauses or for a descendant segment:

```
OBTAIN NEXT record WHERE expression1 [AND expressionN]
```

As indicated by the brackets, this command is also used to pass compound WHERE clauses to IDMS/DB.

IDMS/DB processes the adapter call and returns a record if two conditions are met:

- There is a SELECT path in the LR path-group that can process the particular request.

  Note: For entry segments without WHERE clauses, there must be a null SELECT clause defined for the logical record. If there is no available SELECT path, IDMS/DB returns message 2041, and the adapter terminates its processing with an EDA949 message.

- There is a complete LRF record created by the path-group logic. If the selected path-group can return partial records, the adapter processes returned records until the first partial record is returned and the LR status field is not LR-FOUND or LR-NOT-FOUND. When the adapter encounters any other status in the LR status field, it aborts the record retrieval process and returns messages EDA967 and EDA949. Logical records that allow for retrieval of partial records should not be used with the adapter.

The adapter continues to call IDMS/DB for LRF records that correspond to a segment until IDMS/DB returns LR-NOT-FOUND in the LR status field. Then the adapter retrieves records for other segments, or it terminates retrieval and the answer set is produced.

Overriding DBNAME and DICTNAME

You can dynamically override the DBNAME and DICTNAME parameters within the Access File. This allows you to specify the DBNAME and DICTNAME for all IDMS/DB Master File and Access File pairs during a session by using a SET command, eliminating the need to modify each Access File manually.

To override the DBNAME and DICTNAME, issue the SET commands outlined below. Once the SET commands are issued, the DBNAME and DICTNAME specified will override the same keywords in all Access Files during your session until you end your session or set the DBNAME and DICTNAME to default.

If no SET command is issued, the default behavior is followed.

Reference: Set DBNAME and DICTNAME

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
ENGINE IDMSR SET DBNAME dbname
```

```
ENGINE IDMSR SET DICTNAME dictname
```

where:

dbname

Is the IDMS/DB database name that you want to access.

dictname

Is the IDMS/DB dictionary name that you want to access.

Note: These set commands can be included in any supported server profile.

Syntax: How to Display the Current Settings

To display the settings that are currently in effect, issue the following command from a client application or from a remote procedure:

```
EX EDAEXEC 'ENGINE IDMSR SET ?'
```

Syntax: How to Revert to Original Settings

To revert to original settings in the Access File, issue the following commands in a remote procedure:

```
ENGINE IDMSR SET DBNAME DEFAULT
ENGINE IDMSR SET DICTNAME DEFAULT
```

Syntax: How to Override the DBNAME and DICTNAME in All IDMS/DB Access Files With SYSDIRL

```
ENGINE IDMSR SET DBNAME SYSDIRL
ENGINE IDMSR SET DICTNAME SYSDIRL
```

Customizing the IDMS/DB Environment

Using WebFOCUS, you can customize the IDMS/DB environment by inverting files and joining Master Files.

File Inversion

When you create a Master File, you create a default representation of a hierarchy. Sometimes, however, you may not want to follow the default route to retrieve records. Two such instances might be when:

1. Your IF criteria screen a segment at the bottom of a subtree.
2. You are processing a multi-path report with IF criteria or sort phrases that are not on a common path.

When these situations occur, using WebFOCUS you can specify a new entry segment (root) at execution time for a specific request. This process is called file inversion, because the parent/descendant relationships along the path linking the original root and the new root are reversed; other parent/descendant relationships remain unchanged.

Note: File inversions only change the file views; they do not affect the data.

Syntax: How to Invert a File

```
TABLE FILE filename.field
```

where:

field

May be any field in the new root segment.

For example, to invert the EMPFULL file so that the office segment is the new root, specify the field OFFICE_CODE:

```
TABLE FILE EMPFULL.OFFICE_CODE
```

You can also display a diagram of the inverted file with the CHECK FILE command (include the RETRIEVE option for a subtree diagram):

```
CHECK FILE filename.fieldname PICTURE [RETRIEVE]
```

You cannot invert a Master File if:

- The path linking the old and new roots passes through segments that have a CALC- or index-based relationship.
- The GETOWN parameter in the ACCESS File for a set-based relationship is set to N.
- It is an LRF Master File.

File inversion is a simple solution to two common problems:

- Denied access because the segment is on the wrong sort path.
- Denied access because the field named in an IF test is not on the root path.

Example: Using File Inversion to Solve Sort Path Problems

In addition to solving the sort path problem, file inversion can improve I/O efficiency which, in turn, minimizes production costs.

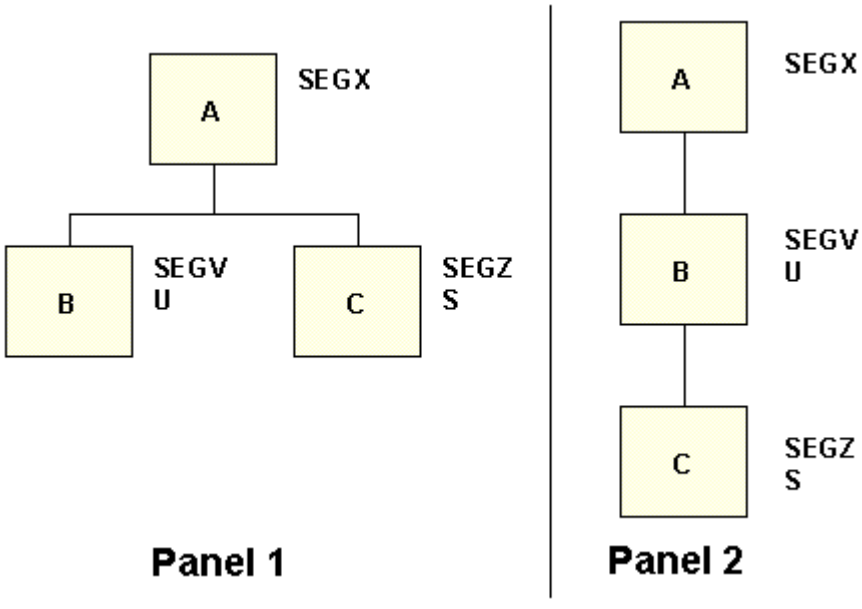
Consider this request:

```
TABLE FILE EMPFULL
LIST SKILL_LEVEL BY SALARY_GRADE
END
```

In panel 1 of the following figure, an error occurs because segments C and B are not on the same path. Therefore, you must use an inverted view:

```
TABLE FILE EMPFULL.SALARY_GRADE
LIST SKILL_LEVEL BY SALARY_GRADE
END
```

In the inverted view (panel 2), segment C is a descendant of segment B. Using this inverted view, the request can be executed.



As this request is executed, record occurrences multiply. Every record of segment C is paired with every record in segment B. If, for example, A had two B descendants and four C descendants, the report would contain eight lines of output. This effect is advantageous when it is necessary to pair every record associated with one linkpath to a record associated with another linkpath. Record pairing may produce undesirable results when the inverted segments are not directly related to each other.

If you use file inversion in conjunction with MISSING=ON, you may access orphan record occurrences that could not be accessed with the default Master File. An orphan record occurrence is one that has no parent record connection. Due to the network structure of IDMS/DB, any hierarchical view may contain orphans. IDMS/DB set connection options OA, OM, or MM indicate the possibility of orphans. Inversion enables the adapter to reconstruct the IDMS/DB relationships, so that these orphans can be retrieved.

Joining Master Files

Using WebFOCUS, you can join the Master Files describing any of these data sources to that of your IDMS/DB data source:

- Other IDMS/DB data sources (SUFFIX=IDMSR)

- VSAM or ISAM or QSAM
- SQL or UDB (Db2)
- DOS/DL1 or IMS
- CA-Datacom/DB
- MODEL 204
- Fixed-format sequential

A JOIN structure is implemented by matching one field that is common to both data sources. The fields on the IDMS/DB target file can be:

- An IDMS/DB CALC field on a network record-type.
- An indexed field (FIELDTYPE=I) on a network record-type.
- A field on an LRF record.

The fields on the host file can be:

- A virtual field located in a host Master File or created as a separate command.
- Any field.

In the Master File, the names of common fields can differ, but their field formats (ACTUAL and USAGE) must be the same.

Syntax: How to Join Two Data Sources

```
JOIN field1 [WITH rfield] IN hostfile [TAG tag1]
TO [ALL] field2 IN crfile [TAG tag2] [AS name]
[END]
```

where:

field1, field2

Are the fields common to both Master Files.

WITH *rfield*

Use only if *field1* is a virtual field; assigns a logical home with a real field in the host file.

hostfile

Is the host Master File.

`TAG tag1`

Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the host file.

The tag name for the host file must be the same in all the JOIN commands of a joined structure.

`ALL`

Use if non-unique relationships exist in the target file.

`crfile`

Is the target or cross-referenced Master File.

`TAG tag2`

Is a tag name of up to eight characters (usually the name of the Master File), which is used as a unique qualifier for fields and aliases in the cross-referenced file. In a recursive joined structure, if no tag name is provided, all field names and aliases are prefixed with the first four characters of the join name.

`AS name`

Assigns a name to the JOIN structure. You must assign a unique name to a join structure if:

- You want to ensure that a subsequent JOIN command will not overwrite it.
- You want to clear it selectively later.
- The structure is recursive, and you do not specify tag names.

`END`

Required when the JOIN command is longer than one line; terminates the command.

To join more than two files as a single structure, indicate the common fields as follows:

```
JOIN field1 IN file1 TO field2 IN file2 AS name1
JOIN field3 IN file1 TO field4 IN file3 AS name2
```

Reference: Usage Notes for the JOIN Command

- Up to 16 joins may be active in one session.
- For a DML target file, field2 must be indexed (FIELDTYPE=I).

- If you intend to use a virtual field as *field1*, specify its field name in the JOIN command and then issue its DEFINE command. Any DEFINE commands issued prior to the JOIN are cleared.
- If you know that the target file is unique, omit the ALL in the JOIN command; omitting ALL reduces I/O overhead.
- To display the JOIN structure, use the CHECK FILE command and specify the name of the host file.

Syntax: How to List JOIN Structures

To list your JOIN structures, enter:

```
? JOIN
```

Syntax: How to Clear JOIN Structures

To clear a specific JOIN structure, specify the name that you assigned to the join:

```
JOIN CLEAR name
```

Syntax: How to Clear All JOIN Structures

To clear all structures, use an asterisk (*) instead of a join name:

```
JOIN CLEAR *
```

Example: Reporting From a Joined Structure

This example joins the DML data source JOBFILE to the IDMS/DB EMPFULL data source based on job codes. First the JOBCODE field in JOBFILE is edited to make it compatible with the JOB_ID field in EMPFULL. The JOIN command is issued prior to the DEFINE. If the DEFINE were issued first, it would be cleared by the JOIN command:

```
JOIN JOBID WITH JOBCODE IN JOBFILE TO
ALL JOB_ID IN EMPFULL AS J1
END
DEFINE FILE JOBFILE
JCODE/A2 = IF JOBCODE LIKE 'A__' THEN '10' ELSE '20';
JOBID/A4 = JCODE|EDIT(JOBCODE,'$99');
END
TABLE FILE JOBFILE
SUM EMP_NAME IN 25 TITLE
BY DEPT_NAME
END
```

The output is:

```
DEPT_NAME               EMP_NAME                    TITLE
---------               --------                    -----
ACCOUNTING AND PAYROLL  RUPERT    JENSON            MGR ACCTNG/PAYROLL
PERSONNEL               ELEANOR   PEOPLES           MGR PERSONNEL
```

Tracing the Adapter for IDMS/DB

From the Web Console main screen, select *Diagnostics* and click *Enable Traces*. The default traces include the Adapter for IDMS/DB trace information.

Chapter 37

Using the Adapter for CA-IDMS/SQL

The Adapter for CA- IDMS/SQL allows applications to access IDMS/SQL data sources. The adapter converts application requests into native IDMS/SQL statements and returns optimized answer sets to the requesting application.

In order to use the Adapter for CA-IDMS/SQL, you must configure and map data to its corresponding server counterparts. Check with your Server Administrator for further information.

Note: For the remainder of this manual, the name IDMS/SQL refers to CA-IDMS/SQL.

In this chapter:

Preparing the IDMS/SQL Environment

Prior to configuring the Adapter for IDMS/SQL using the Web Console, it is necessary to edit the ISTART JCL used to initiate the server. The changes required are documented as follows:

- Allocate the IDMS.LOADLIB and IDMS.DBA.LOADLIB libraries to the server's STEPLIB ddname allocation.
- Allocate ddname SYSCTL to the IDMS.SYSCTL data set.
- Allocate ddname SYSIDMS to the IDMS.SYSIDMS data set.

Configuring the Adapter for IDMS/SQL

Configure the adapter using the Web Console Adapter Add screen and click *Configure*.

In order to connect to an IDMS/SQL database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one IDMS/SQL database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to IDMS/SQL Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Overriding the Default Connection

An SQL session is a connection between the application and the IDMS/SQL data source. It begins when the application connects to a dictionary. You use the CONNECT command to override the IDMS/SQL default (automatic) connection. The length of time an SQL session stays in effect depends on whether the connection began automatically or a CONNECT command was issued. If the CONNECT command was issued, the SQL session is in effect until a COMMIT RELEASE, ROLLBACK RELEASE, or RELEASE command is executed. All of these commands may be executed within the IDMS/SQL session using SQL Passthru. Refer to the appropriate IDMS documentation for more information regarding SQL sessions.

Syntax: How to Override Default Connections With the CONNECT Command

Issue the following syntax within a stored procedure

```
ENGINE SQLIDMS CONNECT TO dictionary
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`dictionary`

Is the data source (dictionary) to start the IDMS/SQL session. The default is the dictionary in effect for the user session. This default is set outside of the server session, for example, with a SYSIDMS DICTNAME parameter. Refer to the appropriate IDMS documentation for a complete description.

Other Session Commands

Other IDMS/SQL commands that affect the IDMS/SQL session can be executed explicitly.

To issue IDMS/SQL session commands such as COMMIT, COMMIT RELEASE, ROLLBACK, ROLLBACK RELEASE, and COMMIT CONTINUE, the syntax is:

```
ENGINE SQLIDMS COMMIT RELEASE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLIDMS SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

The COMMIT setting frees the thread of execution for use by other users. The disadvantage is the cost of repeatedly connecting and acquiring a thread. Threads, once released, may not be available when needed, so you may experience delays while your request waits for a thread.

Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing IDMS/SQL Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IDMS/SQL data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each IDMS/SQL table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for IDMS/SQL

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *IDMS/SQL Data Type Support* on page 992.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for IDMS/SQL synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLIDMS ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=IDMS901,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: TABLENAME=[*location.*][*owner.*]*tablename*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
DBSPACE	Optional keyword that indicates the storage area for the table. For example: *datasource.tablespace* DATABASE *datasource*
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❏ KEYFLD is the FIELDNAME of the common column from the parent table. ❏ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Master Files

The following describes the three types of Master File declarations:

Declaration Type	Description
File	Names the file and describes the type of data source.

Declaration Type	Description
Segment	Identifies a table, file, view, or segment.
Field	Describes the columns of the table or view.

Each declaration must begin on a separate line. A declaration consists of attribute-value pairs separated by commas. A declaration can span as many lines as necessary, as long as no single keyword-value pair spans two lines.

Do not use system or reserved words as names for files, segments, fields, or aliases. Specifying a reserved word generates syntax errors.

Syntax: How to Declare File Attributes

Each Master File begins with a file declaration. The file declaration has two attributes, FILENAME and SUFFIX:

```
FILE[NAME]=file, SUFFIX=SQLIDMS [,$]
```

where:

file

Identifies the Master File. The file name can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents.

SQLIDMS

Is the value for the adapter.

Syntax: How to Declare Segment Attributes

Each table described in a Master File requires a segment declaration. The segment declaration consists of at least two attributes, SEGNAME and SEGTYPE:

```
SEGNAME=segname, SEGTYPE=S0 [,$]
```

where:

segname

Is the segment name that serves as a link to the actual IDMS/SQL table name. It can consist of a maximum of 8 alphanumeric characters. It may be the same as the name chosen for FILENAME, the actual table name, or an arbitrary name.

The SEGNAME value in the Master File must be identical to the SEGNAME value specified in the Access File.

S0

Indicates that IDMS/SQL assumes responsibility for both physical storage of rows and the uniqueness of column values (if a unique index or calc key exists). It always has a value of S0 (S zero).

Syntax: How to Declare Field Attributes

Each row in a table may consist of one or more columns. These columns are described in the Master File as fields with the primary field attributes, FIELDNAME, ALIAS, USAGE, ACTUAL, MISSING.

Note: You can get values for these attributes from the IDMS/SQL schema definition or standard IDMS/SQL dictionary reports.

```
FIELD[NAME]=fieldname, [ALIAS=]sqlcolumn, [USAGE=]display_format,
[ACTUAL=]storage_format [,MISSING={ON|OFF}], $
```

where:

fieldname

Is the unqualified name of the field. This value must be unique within the Master File. The name can consist of a maximum of 48 alphanumeric characters (including any file name and segment name qualifiers and qualification characters you may later prefix to them in your requests). The name must begin with a letter. Special characters and embedded blanks are not recommended. The order of field declarations in the Master File is significant with regard to the specification of key columns. For more information, see *Primary Key* on page 992.

It is not necessary to describe all the columns of the IDMS/SQL table in your Master File.

sqlcolumn

Is the full IDMS/SQL column name (the adapter uses it to generate SQL statements). This value must comply with the same naming conventions described for field names.

display_format

Is the display format. The value must include the field type and length and may contain edit options.

The data type of the display format must be identical to that of the ACTUAL format. For example, a field with an alphanumeric USAGE data type must have an alphanumeric ACTUAL data type.

Fields or columns with decimal or floating-point data types must be described with the correct scale (s) and precision (p). Scale is the number of positions to the right of the decimal point. Precision is the total length of the field.

For the server, the total display length of the field or column includes the decimal point and negative sign. In SQL, the total length of the field or column excludes positions for the decimal point and negative sign.

For example, a column defined as DECIMAL(5,2) would have a USAGE attribute of P7.2 to allow for the decimal point and a possible negative sign.

storage_format

Is the storage format of the IDMS/SQL data type and length, in bytes, for the field.

ON

Displays the character specified by the NODATA parameter for missing data. For more information, see *MISSING Attribute* on page 990.

OFF

Displays blanks or zeroes for fields having no value. OFF is the default value. For more information, see *MISSING Attribute* on page 990.

MISSING Attribute

In a table, a null value represents a missing or unknown value. It is not the same as a blank or a zero. For example, a column specification that allows null values is used where a column need not have a value in every row (such as a raise amount in a table containing payroll data).

Note:

- The default NODATA character is a period (.).
- A column in an IDMS/SQL table that allows null data does not need to include the NULL clause in its table definition, since that is the default. In the Master File for that table, the column that allows null data must be described with the MISSING attribute value ON. The default for this attribute is OFF, which corresponds to the NOT NULL attribute in the IDMS/SQL table definition.
- Null data values appear as zeroes or blanks, if the column allows null data but the corresponding field in the Master File is described with the MISSING attribute value OFF.

Access Files

Each Master File must have a corresponding Access File. The file name of the Access File must be the same as that used for the Master File.

The Access File serves as a link between the server and the data source by providing the means to associate a segment in the Master File with the table it describes. The Access File minimally identifies the table and primary key. It may also indicate the logical sort order of data.

Syntax: How to Establish Segment Declarations

The segment declaration in the Access File establishes the link between one segment of the Master File and the actual IDMS/SQL table or view. Attributes that constitute the segment declaration are:

```
SEGNAME=segname, TABLENAME=[schema.]tablename, [,DBSPACE=storage,]
  [,KEYS={0|n}] [,KEYORDER={ASC|LOW|HIGH|DESC}]  ,$
```

where:

segname

Is the 1- to 8-character SEGNAME value from the Master File.

schema

Is the IDMS/SQL schema name for the table or view. It can consist of a maximum of 8 characters. If it is not specified, IDMS/SQL searches for a temporary table definition for the named table. If the named table does not exist, IDMS/SQL uses the current schema in effect for the current user session.

tablename

Is the name of the table or view. It can consist of a maximum of 18 characters.

Note: If any part of the TABLENAME includes a dollar sign ($), enclose that part in double quotation marks, and enclose the entire TABLENAME value in single quotation marks.

The maximum IDMS/SQL length for a fully qualified tablename is 36. All names must conform to the rules for identifiers stated in the *CA-IDMS/DB Release SQL Reference* manual.

storage

Is the IDMS/SQL segment and area name (in the form segment.area). If not specified, IDMS/SQL uses the default area associated with the schema. Enclose it in double quotation marks if it begins with a number or special character or contains special characters.

The Access File DBSPACE attribute overrides both the SET command and the installation default.

n

Is the number of columns that constitute the primary key. It can be a value between 0 and 64. 0 is the default value. For more information, see *Primary Key* on page 992.

LOW (ASC)

Indicates an ascending primary key sort sequence. LOW is the default value. ASC is a synonym for LOW.

HIGH (DESC)

Indicates a descending primary key sort sequence. DESC is a synonym for HIGH.

Primary Key

The primary key of a table consists of the column or combination of columns whose values uniquely identify each row of the table. In the employee table, for example, every employee is assigned a unique employee identification number. Each employee is represented by one and only one row of the table, and is uniquely identified by that identification number.

The order of field declarations in the Master File is significant to the specification of key columns. To define the primary key in a Master File, describe its component fields immediately after the segment declaration. You can specify the remaining fields in any order. In the Access File, the KEYS attribute completes the process of defining the primary key.

To identify the primary key, the adapter uses the number of columns (n) indicated by the KEYS attribute in the Access File and the first n fields described in the Master File.

Typically, the primary key is supported by the creation of a unique index in the SQL language to prevent the insertion of duplicate key values. The adapter itself does not require any index in the column(s) comprising the primary key (although a unique index is certainly desirable for both data integrity and performance reasons).

IDMS/SQL Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96

Customizing the IDMS/SQL Environment

The Adapter for IDMS/SQL provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Setting the User-specific Schema Name

The Adapter for IDMS/SQL uses the user-specified schema name as the first qualifier for all SQL requests involving SQL tables or views. This command overrides the IDMS/SQL current schema in effect and precludes the specification of unqualified table names. This prevents passing of unqualified table names to IDMS/SQL.

Syntax: How to Set SESSION

```
ENGINE SQLIDMS SET SESSION CURRENT SCHEMA schema
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`schema`

Is the name of the schema in SQL requests.

Controlling Transactions

IDMS/SQL protects data being read by one user from changes (INSERT, UPDATE, or DELETE) made by others. The Isolation Level setting governs the duration of the protection. That is, the Isolation Level determines when shared locks on rows are released, so that those rows or pages become available for updates by other users. IDMS/SQL allows you to dynamically set the Isolation Level within the server session using the IDMS/SQL SET TRANSACTION command.

The SET TRANSACTION CURSOR STABILITY or TRANSIENT READ command affects the duration of row or page shared locks on IDMS/SQL tables for the duration of the IDMS/SQL transaction. You can specify the command within a stored procedure. The setting remains in effect for the server session or until you reset it.

Syntax: How to Control Transactions

```
ENGINE SQLIDMS SET TRANSACTION
 {CURSOR STABILITY|TRANSIENT READ|READ ONLY|READ WRITE}
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`CURSOR STABILITY`

Provides the maximum amount of concurrency while guaranteeing the integrity of the data selected. CURSOR STABILITY is the default value.

`TRANSIENT READ`

Allows the reading of records locked by other users. This is recommended for SQL request only. Transient read prevents the SQL transaction from performing updates. Use this only when you do not need the data retrieved to be absolutely consistent and accurate. If you specify Transient Read, IDMS/SQL assumes it is read-only.

`READ ONLY`

Allows data to be retrieved, but does not allow the data source to be updated.

`READ WRITE`

Allows data to be retrieved, and allows the data source to be updated.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create.

For the duration of the session, the adapter places these tables in the IDMS/SQL tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, IDMS/SQL uses the default tablespace for the connected user.

Syntax: How to Set DBSPACE

```
ENGINE SQLIDMS SET DBSPACE storage
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`storage`

Is the segment.area name that will contain the IDMS/SQL tables created by the CREATE FILE command. If a name is not specified, the table will be placed in the IDMS/SQL default area for the current schema in effect for the user's SQL session.

Note: This command will only affect CREATE TABLE requests made by Table Services. It does not affect Passthru CREATE TABLE commands.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the IDMS/SQL index space implicitly created by the CREATE FILE and HOLD FORMAT SQLIDMS commands.

Syntax: How to Set IXSPACE

```
ENGINE SQLIDMS SET IXSPACE [index-spec]
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`index-spec`

Is the portion (up to 94 bytes) of the SQL CREATE INDEX statement beginning with IN segment.area (as specified in the *CA-IDMS/DB Reference* CREATE INDEX syntax diagram).

Note: To reset to the IDMS/SQL default index space parameters, issue the SET IXSPACE command with no operands.

Example: Specifying Segment.Area Name and Index Block

The following example shows how to set the segment.area name and INDEX BLOCK of the CREATE INDEX statement with IXSPACE:

```
ENGINE SQLIDMS
SET IXSPACE IN EMPSEG.EMPAREA INDEX BLOCK CONTAINS 5 KEYS
END
```

You can use the SQL ? query command to determine the current IXSPACE setting. If the current setting is the default, IXSPACE does not display in the SQL SQLIDMS ? output.

Note: This command will only affect CREATE INDEX requests made by Table Services. It does not affect Passthru CREATE INDEX commands.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLIDMS SET PASSRECS {ON|OFF}
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides an informational message after the successful execution of an SQL Passthru UPDATE or DELETE command. ON is the default value.

`OFF`

Does not provide a message after the successful execution of an SQL Passthru UPDATE or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Note that since, by definition, the successful execution of an INSERT command always affects one record, INSERT does not generate this information.

IDMS/SQL Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Specifying Block Size for Retrieval Processing

The Adapter for IDMS/SQL supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE SQLIDMS SET FETCHSIZE n
```

where:

`SQLIDMS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Chapter 38

Using the Adapter for IMS

The Adapter for IMS allows applications to access IMS data sources. The adapter converts application requests into native IMS statements and returns optimized answer sets to the requesting application.

The Adapter for IMS is supported on the z/OS platform and uses the IMS DBCTL environment to connect to IMS databases.

In this chapter:

IMS Environments: Overview

The Adapter for IMS supports the DBCTL environment to connect to IMS databases.

Advantages of IMS DBCTL

The advantages of using DBCTL include the following:

- **Direct communication between your application and IMS.** Your task establishes its own connection to IMS by loading and calling the DBCTL subsystem when needed. It then issues DL/I calls directly to IMS.
- **Enhanced security.** You have access to standard security systems through the standard SAF interface. The SAF interface is supported by security products such as RACF, CA-TOP SECRET, and CA-ACF2. Before allowing access to a particular PSB, the system verifies that the user is authorized to read the PSB.

- **Dynamic PSB selection and switching within your application.** When you have Access Files, you do not have to issue a command to select a PSB, so the selection is transparent to your application.
- **Sharing of PCBs between applications.** Multiple users can share the same PCBs, reducing the number of duplicate PCBs in a PSB and minimizing overhead.
- **IMS SET ? query command.** DBCTL enables you to view current settings (PSB, IMSSEC, IMSCLASS, IMSMODE, IMSPZP) by issuing the IMS SET ? query command.
- **Read/write capabilities.**
- **Creation of Master and Access files using the Web Console.**

Preparing the IMS Environment

This section describes steps for preparing the IMS DBCTL environment.

Before configuring the Adapter for IMS verify that your site is running an IBM-supported version of IMS and that all APPLCTN macros describing PSBs that will be accessed by multiple users specify SCHDTYP=Parallel. If necessary, modify the APPLCTN macros for such PSBs to include the attribute SCHDTYP=Parallel, and rerun the IMS SYSGEN to make the changes effective.

The Adapter for IMS connects using the DBCTL environment. For the connection to be made, the following libraries must be identified and allocated to the STEPLIB in the IRUNJCL JCL member of your server configuration file:

- The DFSPZP library.
- The SDFSRESL library.

Note: For example, for PDS deployment, the server configuration file is of the form *qualif*.*release*.*server_type*.DATA where *qualif* is provided by the user. The *release* and *server_type* vary with the release and license key being used. Refer to the Server Installation for z/OS in the *Server Installation* manual for further information on the configuration data set naming convention.

The PZP library must have a member that has been generated to identify the correct DBCTL environment for the connection. The following steps must be completed before the server can be configured, using the Web Console or Data Management Console, for the Adapter for IMS:

1. *How to Create the DRA Startup Table: DFSPZPxx*
2. *Assembling and Linking the DRA Startup Table*
3. *Establishing Security*

Syntax: How to Create the DRA Startup Table: DFSPZPxx

The Database Resource Adapter (DRA) is the interface between a user task and DBCTL. The DRA Startup Table contains values that define the characteristics of the DRA. The name of the DRA Startup Table is

```
DFSPZPxx
```

where:

xx

Is a two-character suffix that should be chosen to comply with your site standards.

Once this suffix has been chosen, its value is needed during the configuration phase using the Web Console or Data Management Console.

Example: Assembling and Linking the DRA Startup Table

This sample JCL illustrates how to assemble and link the DFSPZPxx member. You can normally find sample JCL in your IMS installation library.

```
//job card goes here
//ASSEMBLE EXEC  PGM=ASMA90,REGION=2M,PARM='OBJECT,NODECK'
//SYSLIB     DD DSN=IMS.MACLIB,DISP=SHR
//SYSLIN     DD UNIT=SYSDA,DISP=(,PASS),
//               SPACE=(80,(100,100),RLSE),
//               DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT   DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1     DD UNIT=SYSDA,DISP=(,DELETE),
//               SPACE=(CYL,(10,5))
//SYSIN      DD    *
PRP      TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZPxx CSECT
         EJECT
         DFSPRP DSECT=NO,                                              X
               DBCTLID=IMSx,                                           X
               DDNAME=DFSRESLB,                                        X
               DSNAME=IMS.RESLIB,                                      X
               CNBA=150,                                               X
               MAXTHRD=150,                                            X
               MINTHRD=5
         END
/*
//*
//LINK     EXEC  PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,ASSEMBLE),
//                 REGION=4M
//SYSLIN     DD DSN=*.ASSEMBLE.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT   DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1     DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//               SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
//SYSLMOD    DD DISP=SHR,DSN=qualif.outputlib
```

where:

x

Is the IMS version number.

qualif.outputlib

Is the output library where the PZP module (DRA Startup Table) will be stored.

xx

Is the two-character suffix chosen for the DRA Startup Table.

Note: The *qualif.outputlib* library must be concatenated ahead of IMS.RESLIB in the allocation for DDNAME STEPLIB in the IRUNJCL member of your server configuration file.

Reference: **Keywords for Generating the DFSPZP Library**

The following chart describes the keywords required to generate the DFSPZP library. Other keywords that affect your IMS environment, and may be required at your site, are described in the *IBM IMS System Definition Reference*.

Keyword	Description	Default
AGN	Is a 1 to 8 character application group name used as part of the DBCTL security function. For more information on DBCTL security, see the *IMS System Administration Guide*.	N/A
CNBA	Is the total number of Fast Path buffers for the server. For a description of Fast Path DEDB buffer usage, see the *IMS System Administration Guide*. You can omit this parameter if your site does not utilize Fast Path.	N/A
DBCTLID	Is the 4-character name of the DBCTL region. This is the same as the IMSID parameter in the DBC procedure. For more information on the DBC procedure, see the *IBM IMS System Definition Reference*.	SYS1

Keyword	Description	Default
DDNAME	Must be DFSRESLB. It is the DDNAME that will be allocated to the DBCTL RESLIB library (see the DSNAME keyword).	N/A
DSNAME	Is the 1- to 44-character data set name of the DBCTL RESLIB library. It must contain the DRA modules and be MVS authorized.	IMS.RESLIB
FPBOF	Is the number of Fast Path DEDB overflow buffers to be allocated per thread. For a description of Fast Path DEDB buffer usage, see the *IMS System Administration Guide*. You can omit this keyword if your site does not use Fast Path.	00
FPBUF	Is the number of Fast Path DEDB buffers to be allocated and fixed per thread. For a description of Fast Path DEDB buffer usage, see the *IMS System Administration Guide*. You can omit this parameter if your site does not utilize Fast Path.	00
FUNCLV	Is the level of the DRA that the CCTL supports. FUNCLV=1 means the CCTL uses the DRA at the IMS 3.1 level.	1
MAXTHRD	Is the maximum number of concurrent DRA threads. The value cannot exceed 255.	1
MINTHRD	Is the minimum number of concurrent DRA threads available. Since the number of threads specified by this keyword will be allocated at all times, regardless of whether they are used, be careful when selecting this value. The value cannot exceed 255.	1
SOD	Is the output class to use for a SNAP DUMP of abnormal thread termination.	A

Keyword	Description	Default
TIMEOUT	Is the number of seconds a CCTL should wait for the successful completion of a DRA TERM request. Specify this value only if the CCTL is coded to use it. This value is returned to the CCTL upon completion of an INIT request.	60
TIMER	Is the number of seconds between attempts of the DRA to identify itself to DBCTL during an INIT request.	60
USERID	Is the 8-character name of the CCTL region. No two CCTLs (servers accessing the same IMS region through DBCTL) can have the same USERID (address space ID).	N/A

Establishing Security

The DBCTL environment, when accessed through the server, enables the use of security systems through the standard SAF interface. With the SAF interface, your site can use security products such as RACF, CA TOP SECRET, and CA ACF2 to restrict access to PSBs. Before allowing access to a particular PSB, the security system verifies that the user is authorized to read the PSB.

Note: The DBCTL function is tested and verified with the RACF product. Other SAF products using identical calls should perform properly when installed and verified by your site's security administrator.

RACF comes with several predefined security classes. Customer sites can use an existing class (such as PCICSPSB) or define a resource class specifically for DBCTL use.

Syntax: How to Define a PSB Resource

The following syntax illustrates how to define a PSB resource through a PCICSPSB profile to RACF, and how to grant users permission to access the resource.

```
RDEFINE PCICSPSB (psbname) UACC(NONE) NOTIFY(sys_admin_userid)
PERMIT psbname CLASS(PCICSPSB) ID(user_or_group [user_or_group ...])
ACCESS(READ)
```

where:

psbname

Is a PSB name to be protected by RACF.

sys_admin_userid

Is the user ID of the system administrator.

user_or_group

Authorizes the user IDs and/or user groups listed in the PERMIT command to read the specified PSB. Separate items in the list with blanks.

At run time, after the PSB is selected, but prior to scheduling it, the server issues a call to the security system and verifies that the user is authorized to read the PSB.

Configuring the Adapter for IMS

This section describes configuration steps and parameters for the Adapter for IMS in the DBCTL environment.

Procedure: How to Configure the Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Note: Restart the server so the connection to DBCTL can be initialized.

Reference: Connection Attributes for IMS DBCTL

The IMS Adapter in under the *DBMS* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

IMS Sec

ON activates DBCTL security. This setting requires that the server be in authorized mode.

OFF does not implement DBCTL security.

IMS DBCTL

START.

IMS RECONNECT

The adapter uses the IMS DBCTL environment to connect to IMS databases. Reconnection to DBCTL is now handled automatically using the IMS RECONNECT parameter.

The default reconnection time interval is 2 minutes. However, if an ABEND occurs during an IMS shut down, you must increase the reconnection time interval.

Note: For best results, we strongly recommend that you use the IMS /CHECKPOINT FREEZE command as your mechanism for shutting IMS down.

If this parameter is set to 0, reconnection works as in previous releases, without automatic restart. In this case, you must restart the server to establish the connection to DBCTL.

IMS Class

Points to a valid class that contains the security rule. The Systems Administrator can define any class name required by the security implementation of the site. The default value is PCICSPSB.

IMS PZP

The suffix of the IBM/IMS DFSPZP module that is created during the IMS/DBCTL installation process. For more information, see in *How to Create the DRA Startup Table: DFSPZPxx* on page 1001.

Managing IMS Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the IMS data types.

You can create metadata entirely from the Web Console or the Data Management Console. The generated synonyms consist of a Master File and an Access File.

Creating Synonyms

Synonyms define unique names (or aliases) for each IMS PSB that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

For the Adapter for IMS to access an IMS PSB, you must create a synonym for each IMS PSB you access. The logical description of an IMS file is stored in a Master File, which describes the field layout.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for IMS

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Filter PSB Name

Selecting this check box adds a Name input field that allows you to filter the PSB names for which you wish to create synonyms.

Enter a string for filtering the PSB names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select PSBs which begin with the letters ABC; %ABC to select PSBs which end with the letters ABC; %ABC% to select PSBs which contain the letters ABC at the beginning, middle, or end

DBD library name

Is the MVS library name that corresponds to the IMS DBDLIB. This must be a fully qualified MVS data set name.

PSB library name

Is the MVS library name that corresponds to the IMS PSBLIB. This must be a fully qualified MVS data set name.

PSB Selection

Select a member name from the drop-down list.

COBOL FD Selection Options

Map using COBOL FDs

Select this option if you have a Cobol FD library available that describes the PCB view. You have to select segments in this case.

If you do not select this option, the final screen opens without the option to select a Cobol FD entry. Create Synonym then creates a skeleton synonym using the Data Base Definition defined fields.

File System

Select one of the following from the drop-down list:

- *Fully qualified PDS name* to enter a fully qualified MVS Library in the entry box.
- *Absolute HFS directory pathname* to enter the HFS location that contains the COBOL FD.
- *Applications* to enter a pre-configured HFS directory.

PDS name or Directory name

Depending on your selection for File System, enter the fully qualified PDS name or absolute directory path that contains the COBOL FD.

Member Name or File Name

Depending on your selection for File System, enter the member name or file that contains the COBOL FD.

You can enter a string for filtering these names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select names which begin with the letters ABC; %ABC to select names which end with the letters ABC; %ABC% to select names which contain the letters ABC at the beginning, middle, or end.

File Extension

If you selected *Absolute HFS directory pathname* as the file system, enter the extension of the file that contains the COBOL FD.

COBOL FD

For each segment that you want to map using a COBOL FD, select the appropriate COBOL FD from the drop-down list.

Additional Options

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Example: Creating a Synonym

Use the following steps to create a synonym for the AIHDAM data source:

1. From the Web Console menu bar, click *Applications.*

   The Applications page opens.

2. Click the *New* button and select *Synonym* from the drop-down menu.

   The *Select adapter to configure* or *Select connection to create synonym* pane opens.

3. Click *IMS DBCTL*.

   Note: You can also create a synonym from the Adapters page by right-clicking the configured IMS connection and selecting *Create Synonym*.

4. Enter the fully qualified DBD library name in the DBD Library input box. For example:

   ```
   IMS.V7R1M0.B.DBDLIB
   ```

5. Enter the fully qualified PSB library name in the PSB Library input box. For example:

   ```
   IMS.V7R1M0.B.PSBLIB
   ```

6. Check Filter PSB Name and enter the following mask in the text box that displays:

   ```
   ai%
   ```

7. Click *Next*.

   The PSB Selection screen opens.

8. Select the appropriate PSB name from the drop-down list, for example: AIHDPSB.
9. If you have a COBOL FD for the database, check *Map using COBOL FDs*.
10. Click *Next*.

    The PCB Selection screen opens.

11. Select the appropriate PCB, for example, PCB number 5 (which has a secondary index defined).
12. Optionally, edit the name for the synonym in the Default Synonym Name column and select the application under which you want to create this synonym (the default application is baseapp).

13. If you checked *Map using COBOL FDs* in Step 9, enter the information about the COBOL FD for the file system you use for each segment:

    - If you select *Fully qualified PDS name*, enter the name of the PDS that contains the COBOL FD and the member name of the COBOL FD.
    - If you select *Absolute HFS directory pathname* enter the path to the COBOL FD and its file name and extension.
    - If you select *Applications*, choose from the provided HFS directories.

14. Click *Create Synonym*.

    You should get a message indicating that the synonym was created successfully.

15. To view the Master File created, go to the Metadata page, open the application under which you created the synonym, click the synonym name, and select *Edit as Text* from the context menu. This example resulted in the following Master File:

```
FILENAME=AIHDAM, SUFFIX=IMS     , $
  SEGMENT=LANGUAGE, SEGTYPE=S0, $
$  GROUP=AIHDAM_IO, ALIAS=E1, USAGE=A19, ACTUAL=A19, $
    FIELDNAME=EMPL6, ALIAS=EMPL6.HKY, USAGE=I11, ACTUAL=I4, $
    FIELDNAME=LANG6, ALIAS=LANG6.IMS, USAGE=A15, ACTUAL=A15, $
```

16. To view the Access File created, go to the Metadata page, open the application under which you created the synonym, click the synonym name, and select *Edit Access File as Text* from the context menu. This example resulted in the following Master File:

```
PSB=AIHDPSB, WRITE=NO, PCBNUMBER=5, PL1=NO, $
  SEGNAME=LANGUAGE, KEYTYPE=S0, $
      XDFLD=IXEMP6, SRCH=EMPL6, ALTPCBNUMBER=5, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Defining Subsets of DBD Segments and IMS Fields in Segments

The SENSEG parameter is used in a PCB description to define a subset of accessible DBD segments. CREATE SYNONYM produces Master File segments based on the SENSEG information obtained from the PSB, and not from the database definition.

The SENSFLD parameter is used in a PCB description to define a subset of accessible IMS fields in a segment. IMS fields are defined in the DBD. The list of fields in a Master File SEGMENT is generated based on the SENSFLD information, and not on the DBD field definitions.

SENSFLD macros can redefine the layout of an IMS database segment record by using the "START = xx" parameter. Different layouts can be defined in the DBD and SENSFLD. The Master File segment layout (field information) is derived from the SENSFLD.

Master File Attributes

Each Master File that provides access to an IMS data source describes the segments and fields that are available through one IMS PCB.

Note: You do not have to describe every segment from the PCB in the Master File. However, the portion of the hierarchy you describe must be a subtree starting from the root. Any segment or field that you do not describe in the Master File remains invisible to the server.

Reporting costs are largely a function of the volume of data transferred. Therefore, requests issued through the adapter are efficient and cost effective, because only segments referenced in your request are retrieved.

In a Master File, you can define as many segments as WebFOCUS supports as long as the cumulative length of all fields across all segments does not exceed 12,000 bytes (an IMS restriction). See *How to Specify IMS Field Attributes* on page 1015 for additional information about this limit.

Each Master File is stored as a member of a Master File PDS or HFS directory. The member name for a Master File must be the name assigned to that Master File in the Access File record for the corresponding PCB. At run time, the Master File data set is allocated to ddname MASTER.

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.

Syntax: How to Identify Master Files for IMS

Each Master File begins with a file declaration that names the file and describes the type of data source, an IMS data source in this case. The file declaration has two attributes, FILENAME and SUFFIX.

```
FILE[NAME]=name, SUFFIX=IMS [,$]
```

where:

name

Is any 1- to 8-character name.

IMS

Indicates that the Adapter for IMS is required for data retrieval.

Syntax: How to Specify IMS Segment Attributes

Each IMS segment described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE. The SEGNAME value is the name of the corresponding IMS segment. The SEGTYPE value identifies the segment characteristics.

```
SEGNAME=segname, SEGTYPE=segtype [,PARENT=parent] [,$]
```

where:

segname

Is the 1- to 8-character IMS segment name from the NAME parameter of the SENSEG record in the IMS PCB.

segtype

Is S0 or U. U is used to identify unique segments that are data sensitive. A unique segment has no twins (PTR=NOTWIN in the IMS DBD). While a parent segment can have many *types* of unique children, it can have at most one *instance* of each type.

Note: In older releases of the adapter, the SEGTYPE attribute was also used to identify the segment's key type. While those SEGTYPE values are still supported by the adapter, new synonyms created using the Web Console or Data Management Console identify these characteristics using the KEYTYPE attribute in the Access File. For information, see *KEYTYPE in IMS* on page 1033.

parent

Is the name of the parent segment from the IMS data source. Its value comes from the PARENT parameter of the SENSEG record in the IMS PCB.

SEGNAME in IMS

The SEGNAME attribute identifies the IMS segments you can access. The SEGNAME value is the name of the IMS segment from the SENSEG record in the PCB.

The server retrieves segments in top-to-bottom left-to-right sequence as described by the Master File. Therefore, the order of segments in the Master File should be the same as their order in the PCB to maintain the correct hierarchical sequence.

SEGTYPE in IMS

The SEGTYPE attribute is either S0 or U. U is used to identify unique segments that are data sensitive. A unique segment has no twins (PTR=NOTWIN in the IMS DBD). While a parent segment can have many *types* of unique children, it can have at most one *instance* of each type.

Note: In older releases of the adapter, the SEGTYPE attribute was also used to identify the segment key type. While those SEGTYPE values are still supported by the adapter, new synonyms created using the Web Console or Data Management Console identify these characteristics using the KEYTYPE attribute in the Access File. For information, see *KEYTYPE in IMS* on page 1033.

PARENT in IMS

The PARENT attribute identifies the parent of the segment in the hierarchy. It appears in the PARENT parameter of the SENSEG record in the PCB. The only exception is in the root segment, where the PCB either omits the PARENT parameter or specifies PARENT=0. In the Master File, you can specify the PARENT attribute of the root segment as PARENT= , or you can omit it.

Syntax: How to Specify IMS Field Attributes

Each segment consists of one or more fields. The IMS DBD contains FIELD declarations for all sequence (key) and search fields, but other fields are optional.

If the PCB you are describing contains SENFLD records for a segment, the Master File can view only fields explicitly specified in those SENFLD records.

However, if the PCB does not contain any SENFLD records for a segment, you can describe the entire segment in the Master File. You can get information about sequence and search fields from the DBD. To describe other fields, you may have to refer to an external description of the segment, for example, a COBOL FD.

The Master File need not describe all fields from a segment, but it must include an initial subset of the segment (that is, it must start from the beginning and not contain any gaps).

To describe a field in the Master File, you must specify the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. These attributes are discussed in this topic. Note that the adapter does not support the MISSING attribute.

```
FIELD[NAME]=field,[ALIAS=]alias,[USAGE=]display,[ACTUAL=]imsformat ,$
```

where:

field

Is a 1- to 66-character field name. In requests, the field name can be qualified with the Master File and/or segment name. Although the qualifiers and qualification characters do not appear in the Master File, they count toward the 66-character maximum.

alias

Is the alias for a type of field. Possible values are:

imsfield.KEY, the alias for a field that is an IMS key field. Form the alias by appending the suffix KEY to the name of the IMS field.

imsfield.IMS, the alias for a field that is an IMS search field. Form the alias by appending the suffix IMS to the name of the IMS field.

imsfield.HKY, the alias for a field that is the key of the root segment in an HDAM data source. Form the alias by appending the suffix HKY to the name of the IMS field.

display

Is the server display format for the field.

imsformat

Is the server definition of the IMS field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, ,A2, A2,$
```

FIELD NAME in IMS

Field names can consist of a maximum of 66 alphanumeric characters. IMS field names are acceptable values if they meet the following naming conventions:

- Names can consist of letters, digits, and underscore characters. Special characters and embedded blanks are not advised.
- The name must contain at least one letter.
- Duplicate field names (the same field names and aliases) within a segment are not permitted.

Since field names appear as default column titles for reports, select names that are representative of the data.

Note: You can only specify field names in an SQL request. You cannot specify the ALIAS name.

ALIAS in IMS

The ALIAS value in the Master File distinguishes between fields that are defined in the IMS DBD and fields that are not defined to IMS. The adapter uses this information in constructing DL/I calls to IMS.

If a field name in a WHERE clause is a sequence or search field, the adapter may be able to create an SSA that instructs IMS to apply the screening test and return the appropriate records to the server. If the field is not defined in the DBD, the adapter must retrieve all records sequentially from IMS so that the server can screen them.

Note: In certain cases, the adapter can instruct IMS to screen values based on a secondary index. *Segment Redefinition in IMS: The RECTYPE Attribute* on page 1021 describes the technique for taking advantage of a secondary index.

The ALIAS value for a field defined in the DBD is composed of:

- The name of the field as specified in the IMS DBD (1- to 8-characters in length).
- A separation character, the period (.).
- A suffix value that describes whether the field is a sequence field (suffix KEY), a search field (suffix IMS), or the key of the root segment of an HDAM database (suffix HKY).

Except for certain types of control field entries (for example, ORDER and RECTYPE), fields not defined in the DBD should not be assigned ALIAS names. For more information on ORDER and RECTYPE control fields, see *Segment Redefinition in IMS: The RECTYPE Attribute* on page 1021.

USAGE in IMS

The USAGE attribute indicates the display format of the field. An acceptable value must include the field type and length and may contain edit options. The server uses the USAGE format for data display on reports. All standard USAGE formats (A, D, F, I, P) are available.

ACTUAL in IMS

The ACTUAL attribute indicates the server representation of IMS field formats.

For fields defined in the DBD (sequence and search fields), use the format specified in the DBD.

Use the following chart as a guide for describing ACTUAL formats of those fields not defined in the DBD:

COBOL Format	COBOL PICTURE	Bytes of Storage	ACTUAL Format	USAGE Format
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	1	P1	P1
COMP-3	S9V99	2	P2	P5.2
COMP-3	9(4)V9(3)	4	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	4	I4	I7

Note: The USAGE lengths shown are minimum values. You can make them larger and add edit options. You must allow space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

The cumulative length of all fields, across all segments, cannot exceed 12K bytes.

The following example illustrates the DI21PART Master File that provides access to the DI21PART data source.

```
FILE=DI21PART    ,SUFFIX=IMS,$
SEGNAME=PARTROOT ,PARENT=,SEGTYPE=S0,$
  FIELD=PARTKEY  ,ALIAS=PARTKEY.HKY  ,USAGE=A17   ,ACTUAL=A17   ,$
  FIELD=SKIP1    ,ALIAS=             ,USAGE=A33   ,ACTUAL=A33   ,$
SEGNAME=STANINFO ,PARENT=PARTROOT,SEGTYPE=S0,$
  FIELD=STANKEY  ,ALIAS=STANKEY.KEY  ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=SKIP2    ,ALIAS=             ,USAGE=A83   ,ACTUAL=A83   ,$
SEGNAME=STOKSTAT ,PARENT=PARTROOT,SEGTYPE=S0,$
  FIELD=STOCKEY  ,ALIAS=STOCKEY.KEY  ,USAGE=A16   ,ACTUAL=A16   ,$
  FIELD=SKIP3    ,ALIAS=             ,USAGE=A124  ,ACTUAL=A124  ,$
SEGNAME=CYCCOUNT ,PARENT=STOKSTAT,SEGTYPE=S0,$
  FIELD=CYCCKEY  ,ALIAS=CYCCKEY.KEY  ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELD=SKIP4    ,ALIAS=             ,USAGE=A23   ,ACTUAL=A23   ,$
SEGNAME=BACKORDR ,PARENT=STOKSTAT,SEGTYPE=S0,$
  FIELD=BACKEY   ,ALIAS=BACKEY.KEY   ,USAGE=A10   ,ACTUAL=A10   ,$
  FIELD=SKIP5    ,ALIAS=             ,USAGE=A65   ,ACTUAL=A65   ,$
```

Syntax: How to Issue a GROUP Field in IMS

IMS fields can consist of multiple elementary fields. In the Master File, you can break the IMS field into component parts using a GROUP field.

The GROUP record in the Master File describes the combined elementary fields. FIELD records immediately following the GROUP record describe the individual elementary fields.

Each element of the group can have a different format. However, retrieval is more efficient if each USAGE format is of the same type (for example, alphanumeric or packed) as its ACTUAL format; the lengths may differ.

```
GROUP=gname, ALIAS=alias, ELEMENTS=n,$
 FIELD=fld1,,usage1,actual1,$
 .
 .
 .
 FIELD=fldn,,usagen,actualn,$
```

where:

gname

Is the group name. It can be any name that complies with field naming conventions.

alias

Is the IMS field name from the DBD if the group field is a sequence or search field. Possible values are:

imsfield.KEY is the alias for a field that is an IMS key field. Form the alias by appending the suffix KEY to the name of the IMS field.

imsfield.IMS is the alias for a field that is an IMS search field. Form the alias by appending the suffix IMS to the name of the IMS field.

imsfield.HKY is the alias for a field that is the key of the root segment in an HDAM data source. Form the alias by appending the suffix HKY to the name of the IMS field.

Note: The keyword ALIAS is required.

n

Is the number fields in the group.

fld1,...,fldn

Are field names for the individual elements that compose the group.

usage1,...,usagen

Are USAGE formats for the individual elements. For efficient retrieval, each individual USAGE format must be of the same data type as its corresponding ACTUAL format, but their lengths can differ.

actual1,...,actualn

Are ACTUAL formats for the individual elements. For efficient retrieval, each individual USAGE format must be of the same data type as its corresponding ACTUAL format, but their lengths can differ.

Example: Defining a Group Key

The following is an example of a group key definition in the Master File:

```
GROUP=G1, ALIAS=FILEKEY.KEY, ELEMENTS=3 ,$
  FIELD=F1,,A4,A4,$
  FIELD=F2,,P6,P3,$
  FIELD=F3,,I9,I4,$
```

The GROUP in the example describes an IMS key named FILEKEY that is 11 bytes long and consists of a 4-byte alphanumeric field, a 3-byte packed number, and a 4-byte integer.

The ACTUAL length of the GROUP is 11 (4+3+4).

The USAGE length of the GROUP is 16 (4+8+4) because it counts the packed field as 8.

Since the group components are of mixed data types, you must use individual fields in the WHERE expression of the server query.

```
WHERE F1 = ABCD
WHERE F1 BETWEEN A AND B
WHERE F2 = 245
```

If you reference either the group or the first elementary field from the group in your request, the adapter generates qualified SSAs in its DL/I calls for retrieval. Thus, IMS does the screening and returns the segments that pass the screening test back to the server.

However, if you reference an elementary field that is not the first field in the group, the server constructs DL/I calls to retrieve the segments sequentially, and then the server applies the screening test to the returned data.

Note: The adapter does not support a group field within a group field. You may be able to use DEFINE fields in the Master File instead.

Segment Redefinition in IMS: The RECTYPE Attribute

An IMS segment can have multiple definitions. For instance, a segment may contain either shipment or order information, depending on the value of one of its fields. If the field that identifies the type of segment is at the same position and has the same format and length in each redefinition, you can use the RECTYPE attribute to define the different segment types in the Master File.

The record type (RECTYPE) field can be part of the key or of the body of the segment. When you issue a request, you do not have to know which segment definition is called for. The server retrieves the appropriate fields and values based on the value in the RECTYPE field.

In the Master File, you describe the one IMS segment with multiple server segments: a base segment describing the unchanging portion, and a child segment describing each redefinition:

- First define the constant portion of the segment as the base segment; give it the same name as the IMS segment. Include a filler field for the portion that will be redefined. The field that identifies the different types must be in the redefined portion.
- Describe each redefined portion as a child of the base segment. You can give these children any valid segment names, but do not assign them SEGTYPE values. Include a filler field in each child to occupy the positions of fields actually defined in the base segment.

  In each child segment, describe the field that identifies the segment type with FIELDNAME=RECTYPE. The value in the RECTYPE field is the value that identifies the type of segment. For example, the RECTYPE field could contain the value S for a shipment record, or O for an order record. The format of the RECTYPE field can be alphanumeric, integer, or packed.

  Assign the identifying value (for example, S or O) as the ALIAS for the RECTYPE field. If more than one value can identify the same record type, use the ACCEPT attribute instead.

  Describe the remaining fields of each child segment based on its contents and function.

Example: Illustrating an IMS DBD With a Redefined Segment

The following example illustrates an IMS DBD with a redefined segment. The CLIENT segment contains client ID, address, and other client information. The INFO segment contains either shipment information or order information, depending on the value in the INFOTYPE field. If INFOTYPE contains the value S, the segment is a shipment segment. If INFOTYPE contains the value O, the segment is an order segment.

The relevant portions of the DBD are:

```
SEGM NAME=CLIENT,BYTES=(200),PTR=(TWIN),PARENT=0
  FIELD=(CLID,SEQ,U),BYTES=8,START=1,TYPE=C
    .
    .
    .
SEGM NAME=INFO,BYTES=(200),PTR=(TWIN),PARENT=CLIENT
  FIELD=(IKEY,SEQ,U),BYTES=8,START=1,TYPE=C
  FIELD=(INFOTYPE),BYTES=1,START=09,TYPE=C
    .
    .
    .
```

The corresponding Master File represents the IMS INFO segment with three segments:

```
     FILE=IMS1,SUFFIX=IMS
      SEGNAME=CLIENT,SEGTYPE=S0
        FIELD=F1,CLID.KEY,A8,A8,$
1.    SEGNAME=INFO   ,SEGTYPE=S0,PARENT=CLIENT,$
        FIELD=F3,IKEY.KEY,A8,A8,$
        FIELD=,,          A20,A20,$
2.    SEGNAME=SHIP,SEGTYPE=,PARENT=INFO,$
        FIELD=,,          A8,A8,$
        FIELD=RECTYPE,S,  A1,A1,$
        FIELD=SHIPDATE,,  A6,A6,$
          .
          .
          .
        other shipment info
3.    SEGNAME=ORDER,SEGTYPE=,PARENT=INFO,$
        FIELD=,,         A8,A8,$
        FIELD=RECTYPE,O, A1,A1,$
        FIELD=ORDERDATE,,A6,A6,$
          .
          .
          .
        other order info.
```

Note:

1. The base segment is named INFO, like the IMS segment. It contains the key field. The redefined portion is described as a filler field.
2. The SHIP segment describes the shipment record type. The field that corresponds to the IMS INFOTYPE field has FIELDNAME=RECTYPE and ALIAS=S.
3. The ORDER segment describes the order record type. The field that corresponds to the IMS INFOTYPE field has FIELDNAME=RECTYPE and ALIAS=O.

Notice that each child segment has a filler for the key field defined in the base segment.

Syntax: How to Establish Multiple RECTYPE Values in IMS

If multiple values identify the same record type (for example, S or T for a shipment record), use the ACCEPT attribute to enumerate the list or range of acceptable values. In this case, define the ALIAS value as blank.

```
FIELDNAME=RECTYPE, ALIAS=, USAGE=usage, ACTUAL=actual,
   ACCEPT=val1 [OR] val2 [[OR] ...valn],$
```

or

```
FIELDNAME=RECTYPE, ALIAS=, USAGE=usage, ACTUAL=actual,
   ACCEPT=val1 to val2
```

where:

val1,val2,valn

Defines a list or range of values that identifies the record type. A list can be continued on more than one line. Enclose values that contain embedded blanks or special characters within single quotation marks.

The following examples illustrate the ACCEPT attribute:

```
ACCEPT=AAA TO RRR
ACCEPT=136 TO 1029
ACCEPT=RED OR WHITE OR BLUE
ACCEPT=RED WHITE BLUE
ACCEPT=6 OR 11 OR 922 OR 1000
ACCEPT=RED WHITE 'GREEN GREY'
```

Repeating Data in IMS Fields: The OCCURS Segment

In an IMS data source, segments can have repeating fields or repeating groups of fields. The number of repetitions:

- Can be a fixed number.

- Can depend on the value of a field from the parent segment or from the non-repeating portion of the variable segment.
- May have to be calculated from the segment length.

In the Master File, you define multiple segments to describe one IMS variable length segment:

- Define the fixed (single-occurrence) portion of the IMS segment as the base segment. Give it the same name as the IMS segment.
- Define each repeating field or group of fields from the IMS segment as a child segment whose parent is the base segment. The child segment definition has no SEGTYPE, but it includes the OCCURS attribute to specify how many times the field repeats.
- Define the ORDER field in the OCCURS segment if you need to associate a sequence number with each occurrence.

The OCCURS segment is a virtual segment (it does not physically exist) that describes the repetitions to the server. Permissible values for the OCCURS attribute are as follows:

OCCURS=	Description
n	The number of times the field repeats in the segment.
fieldname	The name of a field that contains a value indicating the number of times the field repeats in the segment. The repeating field must be at the end of the segment.
VARIABLE	Indicates that the number of repetitions must be computed from the length of the segment. In this case, the segment must contain a counter field as its first field; the counter field alias in the Master File must be IMS*name*.CNT. The repeating field must be at the end of the segment.

With a fixed number of occurrences, it is possible for the repeating field to be located between other fields in the segment rather than at the end of the segment. In this case, you must define a place-holder field at its position in the base segment. Then, in the OCCURS segment, identify the location of the repeating field by specifying the name of the place-holder field as the POSITION attribute.

Syntax: How to Describe Repetitions to the Server With an OCCURS Segment

```
SEGNAME=occseg, PARENT=imsseg, OCCURS=nfield                ,$
SEGNAME=occseg, PARENT=imsseg, OCCURS=n [,POSITION=posfield]  ,$
SEGNAME=occseg, PARENT=imsseg, OCCURS=VARIABLE              ,$
```

where:

occseg

Is the name of the OCCURS segment. It can be any valid segment name.

imsseg

Is the name of the base segment. It must be the IMS segment name.

nfield

Is the name of a field in the parent or non-repeating portion of the segment whose value is the number of times that the group repeats. You must define this field in the Master File whether or not it is a search field defined in the DBD.

n

Is the fixed number of times that the group repeats in the segment. It is an integer value from 1 to 4095.

posfield

Signals that the repeating field is embedded within the base segment rather than occurring at the end, and names a field in the base segment that marks the starting position of the repeating field.

VARIABLE

Indicates that the length of the repeating segment varies and that the number of occurrences can be computed from each segment. In this case, the (base) segment must contain a counter field as its first field; the counter field's alias value must be

IMS*name*.CNT

where:

IMS*name*

Is the name of the field in the IMS DBD.

In the IMS DBD, a variable length segment differs from a fixed length segment only in the BYTES parameter. For variable length segments, the BYTES parameter consists of two values: the maximum and minimum number of bytes. IMS cannot search for values among the repetitions within a segment. Therefore, in any request that references a field in a repeating group, the server searches and screens the OCCURS segments, not IMS.

Syntax: How to Describe the ORDER Field in IMS

Sometimes the sequence of fields within an OCCURS segment is significant. For example, each instance of the repeating field may represent one quarter of the year, but the segment may not have a field that specifies the quarter to which it applies.

ORDER is an optional counter used to identify the sequence number within a group of repeating fields. Specify it when the order of data is important. The ORDER field does not represent an existing field in the data source; it is used only for internal processing.

The ORDER field must be the last field described in the OCCURS segment.

```
FIELDNAME=name, ALIAS=ORDER, USAGE=In, ACTUAL=I4 ,$
```

where:

name

Is any valid field name.

I*n*

Is an integer format.

Note:

- The ALIAS value must be ORDER.
- The ACTUAL format must be I4.

The ORDER field must be the last field defined in the OCCURS segment.

In requests, you can use the value of the ORDER field. You can also specify a DEFINE statement in the Master File to translate it to more meaningful values. For example:

```
DEFINE QTR/A3 = DECODE ORDER(1 '1ST' 2 '2ND' 3 '3RD' 4 '4TH');
```

A subsequent request could include

```
SELECT TOT.TAXES WHERE QTR=1
```

or:

```
SELECT QTR,BALANCE,INTEREST
```

Example: Issuing OCCURS=n

In the following example, the IMS segment, IMS1, includes a group (consisting of the two fields MONTH and AMOUNT) that repeats 12 times. The COBOL FD for the segment is:

```
01  IMS1
    05  ACCOUNT  PIC X(9)
    05  TYPE     PIC XXX
    05  PAYMENT OCCURS 12 TIMES
        10 MONTH PIC 99
        10 AMT   PIC S9(3)V(99)COMP-3
```

The Master File uses two segments to describe this IMS variable length segment:

```
1. SEGNAME=IMS1,PARENT=,SEGTYPE=S0
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
2. SEGNAME=OCC1,PARENT=IMS1,OCCURS=12
     FIELD=MM, ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
```

1. Segment IMS1 is the base segment. It has the same name as the IMS segment and describes the two non-repeating fields: ACT_NUM and TYPE.
2. The OCCURS segment, OCC1, identifies IMS1 as its parent. It has no SEGTYPE, and it includes the OCCURS attribute. The two repeating fields are described in this segment.

In the following example, the repeating group is not at the end of the segment. It is embedded in the segment before the LNAME field. The COBOL FD for this situation is:

```
01  IMS1
    05  ACCOUNT PIC X(9)
    05  TYPE    PIC XXX
    05  PAYMENT OCCURS 12 TIMES
        10 MONTH PIC 99
        10 AMT   PIC S9(3)V(99)COMP-3
    05  LNAME   PIC X(20)
```

The Master File must include LNAME in the base segment. It must also describe where the repeating fields fit into the base segment by defining a place-holder field before LNAME, equal to the length of the 12 occurrences, and by pointing to the place-holder field with the POSITION attribute:

```
    SEGNAME=IMS1,PARENT=,SEGTYPE=S0
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
1.   FIELD=HOLDIT,ALIAS=,A60,A60,$
     FIELD=LNAME,ALIAS=,A20,A20,$
2.  SEGNAME=OCC1,PARENT=IMS1,OCCURS=12,POSITION=HOLDIT
     FIELD=MM,ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
```

1. The HOLDIT field is the place holder for the repeating group in the base segment (IMS1). Since the repeating group consists of 12 occurrences, each of which is 5 bytes long (A2 and P3, described in segment OCC1), the HOLDIT field is defined as A60.
2. The attribute POSITION=HOLDIT in the OCCURS segment declaration describes where the repeating group is located in the actual (base) segment.

Example: Issuing OCCURS=fieldname

In the next example, the number of occurrences is specified by the value in the TIMES field. The following COBOL FD describes this situation:

```
01  IMS1
    05  ACCOUNT PIC X(9)
    05  TYPE    PIC XXX
    05  TIMES   PIC S999  COMP-3
    05  PAYMENT OCCURS DEPENDING ON TIMES
        10 MONTH PIC 99
        10 AMT   PIC S9(3)V(99)COMP-3
```

The Master File attribute, OCCURS=TIMES, identifies the TIMES field as containing the number of repetitions. The Master File also defines the optional ORDER field as the last field in the OCCURS segment:

```
    SEGNAME=IMS1,SEGTYPE=S0
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
     FIELD=TIMES,ALIAS=,P4,P2,$
1.  SEGNAME=OCC1,PARENT=IMS1,OCCURS=TIMES
     FIELD=MM,ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
2.   FIELD=WHICH,ALIAS=ORDER,I4,I4,$
```

1. The attribute OCCURS=TIMES identifies the value in the TIMES field as the number of instances of the repeating group.
2. The field named WHICH is the optional ORDER field (ALIAS=ORDER). It is an internal counter defined as the last field in the OCCURS segment. It associates a sequence number with each occurrence of the repeating group. With the ORDER field defined, a request can include a test that selects a specific occurrence. For example:

```
   WHERE WHICH = 3
```

Example: Issuing OCCURS=VARIABLE

This example describes a segment in which the number of occurrences must be calculated from the length of the segment. The first field in the segment must be a 2-byte counter field that contains the true length of the segment and is defined to IMS in the DBD. The COBOL FD for this variable length segment is:

```
01  IMS1
    05  COUNTER PIC 99 COMP
    05  ACCOUNT PIC X(9)
    05  TYPE    PIC XXX
    05  PAYMSCHED OCCURS 1 TO 12 TIMES
        10 MONTH PIC 99
        10 AMT   PIC 59(3)V(99)COMP-3
```

The Master File must specify OCCURS=VARIABLE and must describe the counter field with the attribute

```
ALIAS=IMSname.CNT
```

```
    SEGNAME=IMS1,SEGTYPE=S0
1.   FIELD=COUNTFLD,ALIAS=COUNTER.CNT,I2,I2,$
     FIELD=ACT_NUM,ALIAS=ACCOUNT.KEY,A9,A9,$
     FIELD=TYPE,ALIAS=,A3,A3,$
2.  SEGNAME=OCC1,PARENT=IMS1,OCCURS=VARIABLE
     FIELD=MM,ALIAS=,A2,A2,$
     FIELD=AMT,ALIAS=,P6.3,P3,$
```

1. The 2-byte counter field must be the first field in the base segment. Its alias is formed by appending the suffix CNT to the field name defined in the IMS DBD:

   ```
   ALIAS=COUNTER.CNT
   ```

2. In the OCCURS segment definition, the attribute OCCURS=VARIABLE indicates that the first field defined in the Master File contains the segment length and that the number of repetitions must be calculated from this length.

Syntax: How to Redefine Fields in IMS Databases

Support is provided for redefining record fields in IMS databases. This allows a field to be described with an alternate layout.

Within the Master File, the redefined field(s) must be described in a separate unique segment (SEGTYPE=U) using the POSITION=*fieldname* and OCCURS=1 attributes.

```
SEGNAME=segname,SEGTYPE=U,PARENT=parent,
OCCURS=1,POSITION=position ,$
```

where:

segname

Is the segment name.

parent

Is the name of the parent segment.

position

Is the field name of the field being redefined.

A one-to-one relationship is established between the parent record and the redefined segment. The following example illustrates redefinition of the IMS structure described in the COBOL file description:

```
01 ALLFIELDS.
  02 FLD1   PIC X(4).
  02 FLD2   PIC X(4).
  02 RFLD2  PIC 9(5)V99 COMP-3 REDEFINES FLD2.
  02 FLD3   PIC X(8).

FILENAME=REDEF, SUFFIX=IMS,$
  SEGNAME=ONE, SEGTYPE=S0,$
    GROUP=RKEY, ALIAS=KEY  ,USAGE=A4   ,ACTUAL=A4     ,$
      FIELDNAME=FLD1,      ,USAGE=A4   ,ACTUAL=A4     ,$
      FIELDNAME=FLD2,      ,USAGE=A4   ,ACTUAL=A4     ,$
      FIELDNAME=FLD3,      ,USAGE=A8   ,ACTUAL=A8     ,$
    SEGNAME=TWO, SEGTYPE=U, POSITION=FLD2, OCCURS=1, PARENT=ONE ,$
      FIELDNAME=RFLD2,     ,USAGE=P8.2 ,ACTUAL=Z4     ,$
```

The redefined fields may have any user defined name. ALIAS names for redefined fields are not required.

Use of the unique segment with redefined fields helps avoid problems with multipath reporting.

Note:

- Redefinition is a read-only feature and is used only for presenting an alternate view of the data. It is not used for changing the format of the data.
- A field that is being redefined must be equal in length to the field that it is redefining (same actual length).
- For integer and packed fields, you must know your data. Attempts to print numeric fields that contain alpha data will produce data exceptions or errors converting values.

- More than one field can be redefined in a segment.

Access File Attributes

Access Files store database access information used to translate report requests into the required IMS direct calls. For example, the Access File can contain the PSB name, the PCB name, the type of PCB, segment key types, and secondary index information.

In older versions of the adapter, some of these attributes were specified in the Master File instead of the Access File, and the Access File was optional. While those Master Files are still supported by the adapter, new synonyms created using the Web Console or the Data Management Console will have both a Master File and Access File, and the Access File will contain any of these attributes needed for accessing the IMS data source.

An Access File consists of 80-character records in comma-delimited format. A record in an Access File contains a list of attributes and values, separated by commas and terminated with a comma and dollar sign (,$). This list is free-form and spans several lines if necessary. You can specify attributes in any order.

The server reads blank lines, and lines starting with a dollar sign in column one, as comments.

Procedure: How to Select a PSB

You have two options for selecting a PSB within the session:

- You can issue the IMS SET PSB command in your server profile, or in a stored procedure.
- You can identify the PSB in the Access File. Each Access File corresponds to one Master File. It identifies the PSB associated with its corresponding Master File.

If you use an Access File, the adapter identifies the appropriate PSB when you reference a Master File in a request. The selection is automatic and transparent to you.

Syntax: How to Specify Access File Attributes

Each Access File will contain six or more attributes.

```
PSB=psbname,PCBNUMBER= pcb_number,PL1={NO|YES},WRITE={YES|NO},$
SEGNAME=segment_name, KEYTYPE=keytype,[NON_KEYED=NOMATCH,]$
ALTPCBNAME=index_name  , ALTPCBNUMBER=index_PCB_number,$
```

where:

psbname

Is the name of the PSB to use. If the PSB does not exist, the following message is (generated) displayed:

```
(EDA4261) PSB MEMBER NOT FOUND: psbname
```

pcb_number

The position within the PSB that points to the database to be processed.

PL1

Indicates whether the PSB was generated with LANG=(PL/I, COBOL or Assembler). YES indicates PL/I. NO indicates COBOL or Assembler.

WRITE

YES allows SQL update for this database.

NO does not allow SQL update for this database and is the default.

segname_name

Are the segment name(s) defined in the database definition.

keytype

Identifies the segment characteristics.

Possible values are:

S0 indicates that the segment is data sensitive and has no key.

S or S1 indicates that the segment is data sensitive and has a non-unique key.

S2 indicates that the segment is data sensitive and has a unique key.

SH or SH1 indicates that the segment is key sensitive and has a non-unique key.

SH2 indicates that the segment is key sensitive and has a unique key.

NON_KEYED=NOMATCH

Inserts records into a non-keyed child segment without first doing a MATCH on the segment. With this attribute, you can insert a segment instance containing data based on a Master File different from the one used to insert prior instances. Without this attribute, you would get a FOC200 message if the data being inserted did not conform to the original Master File, and the record would not be inserted.

index_name (Optional)

Identifies a secondary index database defined by the 'Procseq=' option within the PSB.

index_PCB_number (Optional)

Identifies the PCB number of the indexed database.

The member name of an Access File must be the same as the member name of the corresponding Master File.

Note: All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following message:

```
(EDA4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```

Reference: KEYTYPE in IMS

The KEYTYPE attribute:

- Identifies the characteristics of the segment key field. A segment can have a unique key, a non-unique key, or no key.

  If a segment has a unique key, at least one FIELD record in the DBD must specify NAME=(fieldname,SEQ,U) or NAME=(fieldname,SEQ). Similarly, if the segment has a non-unique key, at least one FIELD record in the DBD must specify NAME=(fieldname,SEQ,M). If no FIELD record in the DBD specifies a SEQ attribute, the segment has no key.

  The characteristics of the segment key determine the types of SSAs and the number of DL/I calls the adapter must issue to satisfy a retrieval request. To handle keys made up of multiple fields, describe the multiple fields as a group in the Master File (for details, see *How to Issue a GROUP Field in IMS* on page 1019).

- Identifies whether the data from the segment can be retrieved (data sensitive) or if only the segment key is accessible (key sensitive). For a key sensitive segment, the SENSEG record in the PCB includes the parameter PROCOPT=K. Key sensitive segments are used for access to lower-level segments.

The following chart lists the valid KEYTYPE values.

KEYTYPE	Definition	PROCOPT=K In PCB?	NAME= From DBD
`S0`	Data sensitive, no key	No	(name)
`S or S1`	Data sensitive, non-unique key	No	(name,SEQ,M)
`S2`	Data sensitive, unique key	No	(name,SEQ,U) or (name,SEQ)

KEYTYPE	Definition	PROCOPT=K In PCB?	NAME= From DBD
SH or SH1	Key sensitive, non-unique key	Yes	(name,SEQ,M)
SH2	Key sensitive, unique key	Yes	(name,SEQ,U) or (name,SEQ)

Example: Sample Access File for DI21PART

The following is a sample Access File corresponding to the DI21PART Master File.

```
PSB=DI21PSB, WRITE=NO PCBNUMBER=2, PL1=NO,,$
SEGNAME=PARTROOT,  KEYTYPE=S2,$
SEGNAME=STANINFO,  KEYTYPE=S2,$
SEGNAME=STOKCOUNT, KEYTYPE=S2,$
SEGNAME=CYCCOUNT,  KEYTYPE=S2,$
SEGNAME=BAKORDR,   KEYTYPE=S2,$
```

Describing Secondary Indexes in IMS

If you want to access a segment through a field in a segment that is not its sequence field (primary key), IMS provides the option of creating a secondary index on the field. In some situations, using a secondary index improves performance. The secondary index is itself a separate data source. Each of its records contains a value of the field to be indexed and a pointer to the target segment of the data source record containing that value.

When using a secondary index, IMS locates a record by first reading the *index* data source to retrieve the appropriate pointer and then using the pointer to read the *data* source.

If a PCB includes the parameter

PROCSEQ=*index*DBD*name*

the named index is used as the main entry point into the data source.

One Master File and Access File can describe all primary and secondary indexes for a data source. Then, given a request, the adapter can examine all record selection tests to determine the best access path into the data source. The adapter can take advantage of this Auto Index Selection feature if:

- The PSB includes a PCB for each index you may need to access. Each such index PCB must contain a

  PROCSEQ=*index*DBD*name*

parameter that identifies the index DBD.

- The index targets the root segment in the Master File.

In the Master File, prior to the secondary index definitions, you must describe the entire segment of the data source. Every field listed in the DBD is an IMS sequence field or search field, and each secondary index is based on one or more of these fields. You must assign each secondary index field its appropriate alias, as described in *How to Specify IMS Field Attributes* on page 1015.

Note: IMS allows for system-defined sub-sequence fields to make an index unique. The Master File can completely ignore the presence and length of such fields.

Reference: **Using an IMS Secondary Index**

When you create a synonym, information about secondary indexes is placed in the Access File. No auxiliary virtual fields (that is, SKY groups/fields) are created in the Master File.

Note: Existing .sky field suffix-based definitions are supported to ensure compatibility.

Two segments belonging to the same root path might be connected by secondary indexes:

- A target segment has an associated special virtual search field (described in the Access File) to be used in the qualified SSA.
- A source segment has the real base fields to be used in the screening predicates to find the target segment instances.

One target segment may have many secondary indexes based on search fields from different source segments. The secondary index can be based on up to five base fields from the same segment.

Each PCB can be associated with only one index: primary or secondary. A secondary index is defined in the PROCSEQ parameter. When the synonym is created, a primary PCB number is chosen and placed in the PCBNUMBER attribute in the Access File PSB declaration. If the PSB contains several PCBs that define identical hierarchies, the primary PCB number is selected according to following rules:

- If all eligible PCBs have a PROCSEQ parameter, the primary PCB number is taken from the PCB you selected when creating the synonym.

  The root segment is determined, and hierarchical entries processed, in the sequence corresponding to the primary PCB secondary index when an area sweep is performed.
- If at least one of the eligible PCBs does not have a PROCSEQ parameter, the primary PCB number corresponds to the last encountered PCB without a PROCSEQ.

The root is processed in the sequence corresponding to the physical sequence of segment instances for the non-keyed segment, and in the key sequence for key segments when an area sweep is performed.

Sequential numbers of all other PCBs of identical hierarchies associated with the same DBD are stored in the ALTPCBNUMBER attribute in the XDFLD Access File declaration.

Each DBD may be associated with any number of PCBs. Each PCB may have a PROCSEQ parameter to indicate an index key. The segment to which this key refers becomes the tree root segment. Therefore, it is possible to have different hierarchical structures associated with the same database. All identical hierarchies (the only difference is in the PROCSEQ key name) are described by the same Master File. Different Master Files are generated for the various inverted hierarchies.

Access File XDFLD declarations are used to describe secondary indexes. There is a one-to-one relationship between XDFLD statements in the Access File and PROCSEQ parameters in the PCB.

There are two types of XDFLD declarations:

- The first type describes secondary indexing based on PROCSEQ.

  The XDFLD belongs to the tree root segment and must have an ALTPCBNUMBER parameter that refers to the appropriate PCB number. Each PROCSEQ parameter can represent an identical hierarchical structure. Multiple PROCSEQ parameters generate multiple XDFLD declarations in the Access File. Only one such XDFLD statement is needed to build a qualified SSA in the request.

- The second type describes secondary indexing based on the SENSEG INDICES parameter.

  A SENSEG INDICES parameter may be associated with any segment and must not have an ALTPCBNUMBER attribute in the Access File.

  The Access File may contain XDFLD declarations produced from information that is found in the INDICES parameter of the SENSEG macro in a PCB definition. Up to 32 different XDFLD declarations associated with a segment can be built from one SENSEG macro. Any number of such XDFLD declarations might be used to build a qualified SSA in the request. Both types of secondary indexes can be used simultaneously in the qualified SSA in the request. For related information, see *Defining Subsets of DBD Segments and IMS Fields in Segments* on page 1012.

Secondary indexes are physically stored in separate index databases. These databases are referenced by the PROCSEQ and INDICES parameters. One index database can be used to store up to 16 secondary indexes. Such a database, and the related indexes, is referred to as a shared secondary index.

Example: Describing a Shared Secondary Index DBD

You can use information from a shared secondary index DBD to create XDFLD declarations. The following files illustrate the required entries.

DBD and PSB

```
      DBD NAME=DIX1DBD,ACCESS=(HIDAM,VSAM)
      DATASET  DD1=DIX1KAPL,DEVICE=3380
      SEGM     NAME=PARTROOT,PARENT=0,BYTES=50,PTR=T
      LCHILD   NAME=(SEG1,DIX1DBX0),PTR=INDX
      FIELD    NAME=(PARTKEY,SEQ,U),TYPE=C,BYTES=17,START=1
      FIELD    NAME=PARTFIL1,TYPE=C,BYTES=8,START=18
      FIELD    NAME=PARTNAME,TYPE=C,BYTES=23,START=27
      FIELD    NAME=/SX1
      LCHILD   NAME=(X1SEG,DIX1DBX1),PTR=INDX
      XDFLD    NAME=PARTNAMX,SRCH=PARTNAME,SUBSEQ=/SX1,NULLVAL=BLANK
*

      SEGM     NAME=STANINFO,PARENT=PARTROOT,BYTES=85
      FIELD    NAME=(STANKEY,SEQ),TYPE=C,BYTES=2,START=1
      FIELD    NAME=STANFIL1,TYPE=C,BYTES=16,START=3
      FIELD    NAME=STANTYPE,TYPE=C,BYTES=3,START=19
      FIELD    NAME=STANFIL2,TYPE=C,BYTES=26,START=22
      FIELD    NAME=STANASST,TYPE=C,BYTES=4,START=48
      FIELD    NAME=STANFIL3,TYPE=C,BYTES=2,START=52
      FIELD    NAME=STANVALU,TYPE=C,BYTES=2,START=54
*

      SEGM     NAME=STOKSTAT,PARENT=PARTROOT,BYTES=160
      FIELD    NAME=(STOCKEY,SEQ),TYPE=C,BYTES=16,START=1
      FIELD    NAME=STOKFIL1,TYPE=C,BYTES=4,START=17
      FIELD    NAME=STOKNUMS,TYPE=C,BYTES=9,START=21
      FIELD    NAME=STOKFIL2,TYPE=C,BYTES=20,START=30
      FIELD    NAME=STOKFREQ,TYPE=C,BYTES=6,START=50
      LCHILD   NAME=(X2SEG,DIX1DBX2),PTR=INDX
      XDFLD    NAME=CYCLNUMX,SRCH=CYCLNUMB,SEGMENT=CYCCOUNT,        X
               SUBSEQ=/SX1,NULLVAL=BLANK
*

      SEGM     NAME=CYCCOUNT,PARENT=STOKSTAT,BYTES=25
      FIELD    NAME=(CYCLKEY,SEQ),TYPE=C,BYTES=2,START=1
      FIELD    NAME=CYCLNUMB,TYPE=C,BYTES=8,START=3
      FIELD    NAME=CYCLFIL1,TYPE=C,BYTES=4,START=11
      FIELD    NAME=CYCLVALU,TYPE=C,BYTES=7,START=15
      FIELD    NAME=/SX1
*
```

```
      SEGM     NAME=BACKORDR,PARENT=STOKSTAT,BYTES=80
      FIELD    NAME=(BACKKEY,SEQ),TYPE=C,BYTES=10,START=1
      FIELD    NAME=BACKREFR,TYPE=C,BYTES=4,START=77
      DBDGEN
      FINISH
      END


DBPCB01 PCB TYPE=DB,DBDNAME=DIX1DBD,PROCOPT=A,KEYLEN=49
   SENSEG NAME=PARTROOT,PARENT=0
   SENSEG NAME=STANINFO,PARENT=PARTROOT
   SENSEG NAME=STOKSTAT,PARENT=PARTROOT
   SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
   SENSEG NAME=BACKORDR,PARENT=STOKSTAT


DBPCB02 PCB TYPE=DB,DBDNAME=DIX1DBD,PROCOPT=A,KEYLEN=49,              X
               PROCSEQ=DIX1DBX1
   SENSEG NAME=PARTROOT,PARENT=0
   SENSEG NAME=STANINFO,PARENT=PARTROOT
   SENSEG NAME=STOKSTAT,PARENT=PARTROOT
   SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
   SENSEG NAME=BACKORDR,PARENT=STOKSTAT


DBPCB03 PCB TYPE=DB,DBDNAME=DIX1DBD,PROCOPT=A,KEYLEN=49,              X
               PROCSEQ=DIX1DBX2
   SENSEG NAME=STOKSTAT,PARENT=0
   SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
   SENSEG NAME=BACKORDR,PARENT=STOKSTAT
   SENSEG NAME=PARTROOT,PARENT=STOKSTAT
   SENSEG NAME=STANINFO,PARENT=PARTROOT
  PSBGEN LANG=ASSEM,PSBNAME=DIX1PSB,CMPAT=YES
 END
```

Master File:

```
FILENAME=DIX_DIX1DBD, SUFFIX=IMS     , $
  SEGMENT=PARTROOT, SEGTYPE=S0, $
    FIELDNAME=PARTKEY, ALIAS=PARTKEY.KEY, USAGE=A17, ACTUAL=A17, $
    FIELDNAME=PARTFIL1, ALIAS=PARTFIL1.IMS, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FILL1, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=PARTNAME, ALIAS=PARTNAME.IMS, USAGE=A23, ACTUAL=A23, $
    FIELDNAME=FILL2, USAGE=A1, ACTUAL=A1, $


  SEGMENT=STOKSTAT, SEGTYPE=S0, PARENT=PARTROOT, $
    FIELDNAME=STOCKEY, ALIAS=STOCKEY.KEY, USAGE=A16, ACTUAL=A16, $
    FIELDNAME=STOKFIL1, ALIAS=STOKFIL1.IMS, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=STOKNUMS, ALIAS=STOKNUMS.IMS, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=STOKFIL2, ALIAS=STOKFIL2.IMS, USAGE=A20, ACTUAL=A20,$
    FIELDNAME=STOKFREQ, ALIAS=STOKFREQ.IMS, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=FILL1, USAGE=A105, ACTUAL=A105, $
```

```
  SEGMENT=CYCCOUNT, SEGTYPE=S0, PARENT=STOKSTAT, $
    FIELDNAME=CYCLKEY, ALIAS=CYCLKEY.KEY, USAGE=A2, ACTUAL=A2, $
    GROUP=CYCLNUMB, ALIAS=CYCLNUMB.IMS, USAGE=A8, ACTUAL=A8, $
     FIELDNAME=CNMB1, USAGE=A3, ACTUAL=A3, $
     FIELDNAME=CNMB2, USAGE=A3, ACTUAL=A3, $
     FIELDNAME=CNMB3, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=CYCLFIL1, ALIAS=CYCLFIL1.IMS, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=CYCLVALU, ALIAS=CYCLVALU.IMS, USAGE=A7, ACTUAL=A7, $
    FIELDNAME=FILL1, USAGE=A4, ACTUAL=A4, $
```

Access File:

```
PSB=DIX1PSB, WRITE=NO, PCBNUMBER=2, PL1=NO, $
  SEGNAME=PARTROOT, KEYTYPE=S2, $
   XDFLD=PARTNAMX, SRCH=PARTNAME, ALTPCBNUMBER=3, $
   XDFLD=PARTFILX, SRCH=PARTFIL1, ALTPCBNUMBER=4, $
  SEGNAME=STOKSTAT, KEYTYPE=S2, $
   XDFLD=CYCLNUMX, SRCH=CNMB1/CNMB3/CNMB2, BASESEG=CYCCOUNT, $
  SEGNAME=CYCCOUNT, KEYTYPE=S2, $
```

Example: Describing an Inverted Tree Structure

This example describes an inverted tree structure that corresponds to PROSEQ DIX1DBX2 in *Describing a Shared Secondary Index DBD* on page 1037.

Master File:

```
FILENAME=DIX_DIX1DBD, SUFFIX=IMS , $
 SEGMENT=STOKSTAT, SEGTYPE=S0, $
 FIELDNAME=STOCKEY, ALIAS=STOCKEY.KEY, USAGE=A16, ACTUAL=A16, $
 FIELDNAME=STOKFIL1, ALIAS=STOKFIL1.IMS, USAGE=A4, ACTUAL=A4, $
 FIELDNAME=STOKNUMS, ALIAS=STOKNUMS.IMS, USAGE=A9, ACTUAL=A9, $
 FIELDNAME=STOKFIL2, ALIAS=STOKFIL2.IMS, USAGE=A20, ACTUAL=A20, $
 FIELDNAME=STOKFREQ, ALIAS=STOKFREQ.IMS, USAGE=A6, ACTUAL=A6, $
 FIELDNAME=FILL1, USAGE=A105, ACTUAL=A105, $
 SEGMENT=CYCCOUNT, SEGTYPE=S0, PARENT=STOKSTAT, $
 FIELDNAME=CYCLKEY, ALIAS=CYCLKEY.KEY, USAGE=A2, ACTUAL=A2, $
 GROUP=CYCLNUMB, ALIAS=CYCLNUMB.IMS, USAGE=A8, ACTUAL=A8, $
 FIELDNAME=CNMB1, USAGE=A3, ACTUAL=A3, $
 FIELDNAME=CNMB2, USAGE=A3, ACTUAL=A3, $
 FIELDNAME=CNMB3, USAGE=A2, ACTUAL=A2, $
 FIELDNAME=CYCLFIL1, ALIAS=CYCLFIL1.IMS, USAGE=A4, ACTUAL=A4, $
 FIELDNAME=CYCLVALU, ALIAS=CYCLVALU.IMS, USAGE=A7, ACTUAL=A7, $
 FIELDNAME=FILL1, USAGE=A4, ACTUAL=A4, $
 SEGMENT=PARTROOT, SEGTYPE=U, PARENT=STOKSTAT, $
 FIELDNAME=PARTKEY, ALIAS=PARTKEY.KEY, USAGE=A17, ACTUAL=A17, $
 FIELDNAME=PARTFIL1, ALIAS=PARTFIL1.IMS, USAGE=A8, ACTUAL=A8, $
 FIELDNAME=FILL1, USAGE=A1, ACTUAL=A1, $
 FIELDNAME=PARTNAME, ALIAS=PARTNAME.IMS, USAGE=A23, ACTUAL=A23, $
 FIELDNAME=FILL2, USAGE=A1, ACTUAL=A1, $
```

Access File:

```
PSB=DIX1PSB, WRITE=NO, PCBNUMBER=5, PL1=NO, $
 SEGNAME=STOKSTAT, KEYTYPE=S2, $
 XDFLD=CYCLNUMX, SRCH=CNMB1/CNMB3/CNMB2, BASESEG=CYCCOUNT,ALTPCBNUMBER=5, $
 XDFLD=CYCLVALX, SRCH=CYCLVALU, BASESEG=CYCCOUNT, ALTPCBNUMBER=6, $
 SEGNAME=CYCCOUNT, KEYTYPE=S2, $
 SEGNAME=PARTROOT, KEYTYPE=S2, $
 XDFLD=PARTNAMX, SRCH=PARTNAME, $
 XDFLD=PARTFILX, SRCH=PARTFIL1, $
```

This example uses four secondary indexes:

- **PARTNAMX** is the secondary index used to find the segment PARTROOT by the value of the PARTNAME field. This is a SENSEG-type secondary index. Target and source segments are in the same PARTROOT.
- **PARTFILX** is the secondary index used to find the segment PARTROOT by the value of the PARTFIL1 field. This is a SENSEG-type secondary index. Target and source segments are in the same PARTROOT.
- **CYCLNUMX** is the secondary index used to find the segment STOKSTAT by the value of the CNMB1, CNMB3 and CNMB2 fields. This is a PROCSEQ-type secondary index. The target segment is STOKSTAT; the source segment is CYCCOUNT.
- **CYCLVALX** is the secondary index used to find the segment STOKSTAT by the value of the CYCLVALU field. This is PROCSEQ-type secondary index. The target segment is STOKSTAT; the source segment is CYCCOUNT.

Definition description:

1. Secondary index description associated with the target segment.
2. XDFLD parameter defines the secondary index name used to build the SSA.
3. BASESEG parameter defines the source segment name (if it is not the target segment).
4. SRCH parameter defines the names of one or more (up to five) base fields, and their order in the secondary index.

Note: A qualified SSA is built for the target segment if at least one eligible screening condition is provided for the associated primary key, search field, or secondary key base source field.

Example: The PATDB01 Sample Database

The PATDB01 database has a primary index because it is a HIDAM database. It is also defined with three secondary indexes.

Five DBDs are associated with PATDB01: the DBD for the database, the DBD for the primary index, and one DBD for each of the three secondary indexes.

Database DBD for PATDB01

PATDB01 DBD:

```
  PRINT NOGEN
  DBD      NAME=PATDB01,ACCESS=(HIDAM,VSAM)
  DATASET  DD1=PATDB01,DEVICE=3380,BLOCK=4096,SCAN=5
*

  SEGM NAME=PATINFO,PTR=H,PARENT=0,BYTES=233
    FIELD NAME=(SSN,SEQ,U),BYTES=9,START=26,TYPE=C
    LCHILD NAME=(SEGIX,PATDBIX),PTR=INDX
    FIELD NAME=SEQFIELD,BYTES=6,START=1,TYPE=C
    FIELD NAME=REVSEQ,BYTES=6,START=7,TYPE=C
    FIELD NAME=SSNALPHA,BYTES=9,START=35,TYPE=C
    FIELD NAME=EMPID,BYTES=12,START=44,TYPE=C
    FIELD NAME=LNAME,BYTES=12,START=56,TYPE=C
    FIELD NAME=FNAME,BYTES=12,START=68,TYPE=C
    FIELD NAME=ADMDATE,BYTES=8,START=89,TYPE=C
    FIELD NAME=PATID,BYTES=10,START=176,TYPE=C
    FIELD NAME=/SX1
*

    LCHILD NAME=(SEGIX1,PATDBIX1),PTR=INDX
    XDFLD  NAME=IXNAME,SRCH=(LNAME,FNAME),                              X
               SUBSEQ=/SX1,NULLVAL=BLANK
*

    LCHILD NAME=(SEGIX2,PATDBIX2),PTR=INDX
    XDFLD  NAME=IXCOMP,SRCH=(SSNALPHA,EMPID,LNAME)
*

    LCHILD NAME=(SEGIX3,PATDBIX3),PTR=INDX
    XDFLD  NAME=IXADMD,SRCH=(ADMDATE),                                  X
               SUBSEQ=/SX1,NULLVAL=BLANK
*

  DBDGEN
  FINISH
  END
```

Primary Index DBD for PATDB01

PATDBIX DBD:

```
    PRINT NOGEN
    DBD      NAME=PATDBIX,ACCESS=INDEX
    DATASET  DD1=PATDBIX,DEVICE=3380
  *
    SEGM NAME=SEGIX,PARENT=0,BYTES=9
      FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1,TYPE=C
      LCHILD NAME=(PATINFO,PATDB01),INDEX=SSN
    DBDGEN
    FINISH
    END
```

Secondary Index DBDs for PATDB01

PATDBIX1 DBD:

```
  PRINT NOGEN
  DBD      NAME=PATDBIX1,ACCESS=INDEX
  DATASET  DD1=PATDBIX1,DEVICE=3380
*
  SEGM NAME=SEGIX1,PARENT=0,BYTES=28
    FIELD NAME=(IXNAMEIX,SEQ,U),BYTES=28,START=1
    LCHILD NAME=(PATINFO,PATDB01),INDEX=IXNAME,PTR=SNGL
  DBDGEN
  FINISH
  END
```

PATDBIX2 DBD:

```
  PRINT NOGEN
  DBD      NAME=PATDBIX2,ACCESS=INDEX
  DATASET  DD1=PATDBIX2,DEVICE=3380
*
  SEGM NAME=SEGIX2,PARENT=0,BYTES=33
    FIELD NAME=(IXCOMPIX,SEQ,U),BYTES=33,START=1
    LCHILD NAME=(PATINFO,PATDB01),INDEX=IXCOMP,PTR=SNGL
  DBDGEN
  FINISH
  END
```

PATDBIX3 DBD:

```
  PRINT NOGEN
  DBD      NAME=PATDBIX3,ACCESS=INDEX
  DATASET  DD1=PATDBIX3,DEVICE=3380
*
  SEGM NAME=SEGIX3,PARENT=0,BYTES=12
    FIELD NAME=(IXADMDIX,SEQ,U),BYTES=12,START=1
    LCHILD NAME=(PATINFO,PATDB01),INDEX=IXADMD,PTR=SNGL
  DBDGEN
  FINISH
  END
```

PSB to Access PATDB01

```
PCB       TYPE=TP,MODIFY=YES,EXPRESS=YES
PCB       TYPE=TP,EXPRESS=NO,MODIFY=YES,SAMETRM=YES
*

PCB       TYPE=DB,DBDNAME=PATDB01,PROCOPT=GO,KEYLEN=9
SENSEG    NAME=PATINFO,PARENT=0
*

PCB       TYPE=DB,DBDNAME=PATDB01,PROCOPT=GO,KEYLEN=28,PROCSEQ=PATDBIX1
SENSEG    NAME=PATINFO,PARENT=0
*

PCB       TYPE=DB,DBDNAME=PATDB01,PROCOPT=GO,KEYLEN=33,PROCSEQ=PATDBIX2
SENSEG    NAME=PATINFO,PARENT=0
*
PCB       TYPE=DB,DBDNAME=PATDB01,PROCOPT=GO,KEYLEN=12,PROCSEQ=PATDBIX3
SENSEG    NAME=PATINFO,PARENT=0
*
PSBGEN    LANG=COBOL,PSBNAME=TSTPSB01,CMPAT=YES
END
```

This PSB is member TSTPSB01 in the PSB data set because PSBNAME=TSTPSB01.

PATDB01 Master File

```
FILENAME=PATDB01, SUFFIX=IMS     , $
  SEGMENT=PATINFO, SEGTYPE=S0, $
$  GROUP=PATDB01, ALIAS=E1, USAGE=A235, ACTUAL=A235, $
   GROUP=PATINFO, ALIAS=E2, USAGE=A235, ACTUAL=A235, $
    FIELDNAME=SEQFIELD, ALIAS=SEQFIELD.IMS, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=REVSEQ, ALIAS=REVSEQ.IMS, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=SALARY, ALIAS=E5, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=OT_HR_PAY, ALIAS=E6, USAGE=A5, ACTUAL=A5, $
    FIELDNAME=SSN, ALIAS=SSN.KEY, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=SSNALPHA, ALIAS=SSNALPHA.IMS, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=EMPLOYEEID, ALIAS=EMPID.IMS, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=LAST_NAME, ALIAS=LNAME.IMS, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FIRST_NAME, ALIAS=FNAME.IMS, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=DATE_OF_BIRTH, ALIAS=E12, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=RACE, ALIAS=E13, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=ADMIT_DATE, ALIAS=ADMDATE.IMS, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=ADMIT_TYPE, ALIAS=E15, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=DISPOSITION, ALIAS=E16, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=TRANSFER_DATE, ALIAS=E17, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=ALLERGY1, ALIAS=E18, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=ALLERGY2, ALIAS=E19, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=ALLERGY3, ALIAS=E20, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=ALLERGY4, ALIAS=E21, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=HOUSING, ALIAS=E22, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=RPR, ALIAS=E23, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=URIN, ALIAS=E24, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=PRACTITIONAR, ALIAS=E25, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=SHIFT, ALIAS=E26, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=PATINET_ID, ALIAS=PATID.IMS, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=WHO_ADDED, ALIAS=E28, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=DATE_ADDED, ALIAS=E29, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=WHO_EDITED, ALIAS=E30, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=DATE_EDITED, ALIAS=E31, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=STATION_ID, ALIAS=E32, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=DIABETIC, ALIAS=E33, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=DIALYSIS, ALIAS=E34, USAGE=A1, ACTUAL=A1, $
```

```
   GROUP=IXNAME, ALIAS=IXNAME.SKY, USAGE=A24, ACTUAL=A24, $
    FIELDNAME=IX_LNAME, ALIAS=LAST_NAME, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=IX_FNAME, ALIAS=FIRST_NAME, USAGE=A12, ACTUAL=A12, $
   GROUP=IXCOMP, ALIAS=IXCOMP.SKY, USAGE=A33, ACTUAL=A33, $
    FIELDNAME=IX_SSNALPHA, ALIAS=SSNALPHA, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=IX_EMPID, ALIAS=EMPLOYEEID, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=IX_LNAME1, ALIAS=LAST_NAME, USAGE=A12, ACTUAL=A12, $
   GROUP=IXADMD, ALIAS=IXADMD.SKY, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=IX_ADMDATE, ALIAS=ADMIT_DATE, USAGE=A8, ACTUAL=A8, $
SEGMENT=SEG2, SEGTYPE=U, PARENT=PATINFO, OCCURS=1, POSITION=SSNALPHA, $
  GROUP=SSNN, ALIAS=E35, USAGE=A9, ACTUAL=A9, $
  GROUP=SSNUMERIC, ALIAS=E36, USAGE=A9, ACTUAL=A9, $
   FIELDNAME=N1_3, ALIAS=E37, USAGE=A3, ACTUAL=A3, $
   FIELDNAME=N4_5, ALIAS=E38, USAGE=A2, ACTUAL=A2, $
   FIELDNAME=N6_9, ALIAS=E39, USAGE=A4, ACTUAL=A4, $
```

PATDB01 Access File

```
PSB=PATDBPSB, WRITE=NO, PCBNUMBER=4, PL1=NO, $
  SEGNAME=PATINFO, KEYTYPE=S0, $
     XDFLD=IXNAME, SRCH=LAST_NAME/FIRST_NAME, ALTPCBNUMBER=5, $
     XDFLD=IXCOMP, SRCH=SSNALPHA/EMPLOYEEID/LAST_NAME, ALTPCBNUMBER=6, $
     XDFLD=IXADMD, SRCH=ADMIT_DATE, ALTPCBNUMBER=7, $
```

Describing a Partition to the Server

IMS limits the size of its data sources. A site that needs a larger data source may be able to create several smaller data sources by partitioning the large data source based on root key values.

Using extended Access File attributes, you can describe a partition to the server, describe how to concatenate the parts, and assign a name to the concatenated PCB. When you issue a request, you can report from the concatenated PCB name or from any of the individual partitions, depending on the file name you reference in the request.

Syntax: How to Describe a Partition in the Access File

A partition assigns each record to a specific data source depending on its root key value. The first partition contains records with the lowest key values. The next partition contains records with higher key values, and so on. The last partition contains records with the highest key values. Each partition is a separate IMS data source and, therefore, has a separate PCB in the PSB.

To describe the key range in each partition to the server, add the LOWVALUE and HIGHVALUE attributes to the appropriate PCB records in the Access File.

```
PCBNAME=mfdname,PCBTYPE=DB,LOWVALUE={value1|0},HIGHVALUE={value2|FF},$
```

where:

mfdname

Is the name of the Master File for one partition of the large data source. Since the partition is a separate data source with its own DBD, it needs its own PCB and Master File.

value1

Is the lowest key value, in alphanumeric format, in the partition accessed with Master File *mfdname*. 0 is the default value.

value2

Is the highest key value, in alphanumeric format, in the partition accessed with Master File *mfdname*. Hexadecimal FF is the default value.

Note:

- The LOWVALUE and HIGHVALUE attributes are ignored if you do not define a corresponding concatenation of the PCBs.
- The number of partitions you can define for a data source is limited by the number of PCBs in the PSB.
- The partitioning field must be the key of the root segment.
- The partitioning field must have an alphanumeric format.

If the partitioning field is a group composed of multiple subfields, each subfield value must be alphanumeric, and you must specify the concatenated subfield values in the LOWVALUE and HIGHVALUE attributes. For example, if the root key low value is composed of F1=AAAA, F2=88, and F3=BBB, then LOWVALUE=AAAA88BBB.

Describing a Concatenated PCB

In an Access File, you can concatenate individual PCBs by assigning a name to the concatenation and issuing a request against it.

The PCBs that you concatenate do not have to be partitioned, that is, their Access File records do not have to include the LOWVALUE and HIGHVALUE attributes. However, if they do include the partitioning information, the adapter can examine the request and determine which PCBs satisfy the request. Without the partitioning information, the adapter must access every PCB that participates in the concatenation.

Syntax: **How to Describe a Concatenated PCB in the Access File**

The format of the Access File for the Adapter for IMS to provide extended support for concatenated databases is as follows:

1. The CONCATPCB statements provide the PCB number and the low and high value range for the root segment key.

   For example:

```
CONCATPCB=4,LOWVALUE=0,HIGHVALUE=0, $
CONCATPCB=6,LOWVALUE=1,HIGHVALUE=1, $
.............................................................
CONCATPCB=12,LOWVALUE=9,HIGHVALUE=9, $
```

2. The CONCPOSITION statement provides root segment key substring information. The substring value is compared with the key ranges defined in the CONCATPCB statements. Positions in the key are counted from 0 for the substring operation.

The key value is converted to alphanumeric format prior to the substring operation. The key format is taken from the ACTUAL description. A Packed or Zoned format number is processed as an unsigned one.

Conversion rules for the Packed format:

- Result buffer size for Pn is n*2-1 bytes.
- Converted number is adjusted to the right in the buffer with the leading 0.

Conversion rules for the Integer format:

- Only I1, I2, and I4 formats are supported.
- Result buffer size for I1 is 3 bytes; I2 is 5 bytes; I4 is 10 bytes.

For example:

```
CONCPOSITION=1,LENGTH=1, $
```

Example: **Describing a HDAM Concatenated PCB**

```
FILENAME=HDAM0, SUFFIX=IMS     , $
  SEGMENT=LANGUAGE, SEGTYPE=S0, $
    FIELDNAME=EMPL6, ALIAS=EMPL6.HKY, USAGE=P8,  ACTUAL=P4, $
    FIELDNAME=LANG6,  ALIAS=LANG6.IMS, USAGE=A15, ACTUAL=A15, $

PSB=PIHDPSB,PCBNUMBER= 4,PL1=NO,WRITE=NO,$
SEGNAME=LANGUAGE, KEYTYPE=S1  ,$
CONCATPCB=4,LOWVALUE=1,HIGHVALUE=1, $
CONCATPCB=6,LOWVALUE=2,HIGHVALUE=2, $
CONCATPCB=8,LOWVALUE=3,HIGHVALUE=3, $
CONCPOSITION=4,LENGTH=1, $
```

Note: PCBNUMBER in the PSB statement refers to the PCB associated with the root segment.

In the following request, we associate the 5th digit in the key (offset 4 in the key buffer) with the PCB to schedule.:

```
TABLE FILE HDAM0
PRINT *
WHERE EMPL6 EQ 9336
END
```

PCB number 8 will be used for the numeric qualifier 9336. The extracted value is 3 (since packed number 9336 is unpacked to the 7 bytes buffer and the result value is 0009336).

Example: Describing a HIDAM Concatenated PCB

EMPDB01, EMPDB02, and EMPDB03 are HIDAM data sources that illustrate partitioning and concatenation of PCBs in the Access File. The following topics illustrate:

- The EMPDB01, EMPDB02, and EMPDB03 DBDs for the three partitions of the data source, and, since these are HIDAM data sources, the EMPDBIX1, EMPDBIX2, and EMPDBIX3 DBDs for the index data sources.
- A PSB with three PCBs, each corresponding to one partition of the data source.
- The EMPDB01, EMPDB02, and EMPDB03 Master Files for the individual partitions, and the EMPDBJ Master File for the concatenation.
- The Access File corresponding to the PSB. It includes the LOWVALUE and HIGHVALUE attributes for each PCB and the CONCATNAME record to define the concatenation.

EMPDB01 DBD

Data source DBD:

```
PRINT NOGEN
DBD      NAME=EMPDB01,ACCESS=(HIDAM,VSAM)
DATASET  DD1=EMPDB01,DEVICE=3380,BLOCK=4096,SCAN=5
  *
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX1,EMPDBIX1),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
  *
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD      NAME=EMPDBIX1,ACCESS=INDEX
DATASET  DD1=EMPDBIX1,DEVICE=3380
  *
SEGM NAME=SEGIX1,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB01),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

EMPDB02 DBD

Database DBD:

```
PRINT NOGEN
DBD      NAME=EMPDB02,ACCESS=(HIDAM,VSAM)
DATASET  DD1=EMPDB02,DEVICE=3380,BLOCK=4096,SCAN=5
  *
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX2,EMPDBIX2),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
  *
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD      NAME=EMPDBIX2,ACCESS=INDEX
DATASET  DD1=EMPDBIX2,DEVICE=3380
  *
SEGM NAME=SEGIX2,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB02),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

EMPDB03 DBD

Database DBD:

```
PRINT NOGEN
DBD      NAME=EMPDB03,ACCESS=(HIDAM,VSAM)
DATASET  DD1=EMPDB03,DEVICE=3380,BLOCK=4096,SCAN=5
  *
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX3,EMPDBIX3),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
  *
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD      NAME=EMPDBIX3,ACCESS=INDEX
DATASET  DD1=EMPDBIX3,DEVICE=3380
  *
SEGM NAME=SEGIX3,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB03),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

PSB to Access the EMPDB Data Sources

```
PCB       TYPE=TP,MODIFY=YES,EXPRESS=YES
PCB       TYPE=TP,EXPRESS=NO,MODIFY=YES,SAMETRM=YES
PCB       TYPE=DB,DBDNAME=EMPDB01,PROCOPT=GO,KEYLEN=9
SENSEG    NAME=EMPINFO,PARENT=0
PCB       TYPE=DB,DBDNAME=EMPDB02,PROCOPT=GO,KEYLEN=9
SENSEG    NAME=EMPINFO,PARENT=0
PCB       TYPE=DB,DBDNAME=EMPDB03,PROCOPT=GO,KEYLEN=9
SENSEG    NAME=EMPINFO,PARENT=0
PSBGEN LANG=COBOL,PSBNAME=EMPPSBJ,CMPAT=YES
END
```

Master Files to Access the EMPDB Data Sources

The EMPDB01, EMPDB02, and EMPDB03 Master Files each correspond to a PCB for one individual partition of the data source. The EMPDBJ Master File corresponds to the concatenation of the three partitions.

EMPDB01 Master File

```
FILE=EMPDB01,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S0,$
FIELD=SEQFIELD      ,ALIAS=SEQFIELD.IMS    ,USAGE=P6   ,ACTUAL=P2 ,$
FIELD=REVSEQ        ,ALIAS=REVSEQ.IMS      ,USAGE=A6   ,ACTUAL=A6 ,$
FIELD=SALARY        ,ALIAS=                ,USAGE=P8   ,ACTUAL=P4 ,$
FIELD=OT_HR_PAY     ,ALIAS=                ,USAGE=P5   ,ACTUAL=P2 ,$
FIELD=SSN           ,ALIAS=SSN.IMS         ,USAGE=P9   ,ACTUAL=P3 ,$
FIELD=SSNALPHA      ,ALIAS=SSNALPHA.KEY    ,USAGE=A9   ,ACTUAL=A9 ,$
FIELD=EMPID         ,ALIAS=EMPID.IMS       ,USAGE=A12  ,ACTUAL=A12,$
FIELD=ELAST_NAME    ,ALIAS=ELNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=EFIRST_NAME   ,ALIAS=EFNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=RACE          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=ADMIT_DATE    ,ALIAS=ADMDATE.IMS     ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ADMIT_TYPE    ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DISPOSITION   ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=TRANSFER_DATE ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ALLERGY1      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY2      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY3      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY4      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=HOUSING       ,ALIAS=                ,USAGE=A03  ,ACTUAL=A03,$
FIELD=RPR           ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=URIN          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PRACTITIONAR  ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=SHIFT         ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PATINET_ID    ,ALIAS=                ,USAGE=P12  ,ACTUAL=P4 ,$
FIELD=WHO_ADDED     ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_ADDED    ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=WHO_EDITED    ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_EDITED   ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=STATION_ID    ,ALIAS=                ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DIABETIC      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DIALYSIS      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
```

EMPDB02 Master File

```
FILE=EMPDB02,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S0,$
FIELD=SEQFIELD      ,ALIAS=SEQFIELD.IMS    ,USAGE=P6   ,ACTUAL=P2 ,$
FIELD=REVSEQ        ,ALIAS=REVSEQ.IMS      ,USAGE=A6   ,ACTUAL=A6 ,$
FIELD=SALARY        ,ALIAS=                ,USAGE=P8   ,ACTUAL=P4 ,$
FIELD=OT_HR_PAY     ,ALIAS=                ,USAGE=P5   ,ACTUAL=P2 ,$
FIELD=SSN           ,ALIAS=SSN.IMS         ,USAGE=P9   ,ACTUAL=P3 ,$
FIELD=SSNALPHA      ,ALIAS=SSNALPHA.KEY    ,USAGE=A9   ,ACTUAL=A9 ,$
FIELD=EMPID         ,ALIAS=EMPID.IMS       ,USAGE=A12  ,ACTUAL=A12,$
FIELD=ELAST_NAME    ,ALIAS=ELNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=EFIRST_NAME   ,ALIAS=EFNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=RACE          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=ADMIT_DATE    ,ALIAS=ADMDATE.IMS     ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ADMIT_TYPE    ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DISPOSITION   ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=TRANSFER_DATE ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ALLERGY1      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY2      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY3      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY4      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=HOUSING       ,ALIAS=                ,USAGE=A03  ,ACTUAL=A03,$
FIELD=RPR           ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=URIN          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PRACTITIONAR  ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=SHIFT         ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PATINET_ID    ,ALIAS=                ,USAGE=P12  ,ACTUAL=P4 ,$
FIELD=WHO_ADDED     ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_ADDED    ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=WHO_EDITED    ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_EDITED   ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=STATION_ID    ,ALIAS=                ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DIABETIC      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DIALYSIS      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
```

EMPDB03 Master File

```
FILE=EMPDB03,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S0,$
FIELD=SEQFIELD      ,ALIAS=SEQFIELD.IMS    ,USAGE=P6   ,ACTUAL=P2 ,$
FIELD=REVSEQ        ,ALIAS=REVSEQ.IMS      ,USAGE=A6   ,ACTUAL=A6 ,$
FIELD=SALARY        ,ALIAS=                ,USAGE=P8   ,ACTUAL=P4 ,$
FIELD=OT_HR_PAY     ,ALIAS=                ,USAGE=P5   ,ACTUAL=P2 ,$
FIELD=SSN           ,ALIAS=SSN.IMS         ,USAGE=P9   ,ACTUAL=P3 ,$
FIELD=SSNALPHA      ,ALIAS=SSNALPHA.KEY    ,USAGE=A9   ,ACTUAL=A9 ,$
FIELD=EMPID         ,ALIAS=EMPID.IMS       ,USAGE=A12  ,ACTUAL=A12,$
FIELD=ELAST_NAME    ,ALIAS=ELNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=EFIRST_NAME   ,ALIAS=EFNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=RACE          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=ADMIT_DATE    ,ALIAS=ADMDATE.IMS     ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ADMIT_TYPE    ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DISPOSITION   ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=TRANSFER_DATE ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ALLERGY1      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY2      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY3      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY4      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=HOUSING       ,ALIAS=                ,USAGE=A03  ,ACTUAL=A03,$
FIELD=RPR           ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=URIN          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PRACTITIONAR  ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=SHIFT         ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PATINET_ID    ,ALIAS=                ,USAGE=P12  ,ACTUAL=P4 ,$
FIELD=WHO_ADDED     ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_ADDED    ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=WHO_EDITED    ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_EDITED   ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=STATION_ID    ,ALIAS=                ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DIABETIC      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DIALYSIS      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
```

EMPDBJ Master File

```
FILE=EMPDBJ,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S0,$
FIELD=SEQFIELD      ,ALIAS=SEQFIELD.IMS    ,USAGE=P6   ,ACTUAL=P2 ,$
FIELD=REVSEQ        ,ALIAS=REVSEQ.IMS      ,USAGE=A6   ,ACTUAL=A6 ,$
FIELD=SALARY        ,ALIAS=                ,USAGE=P8   ,ACTUAL=P4 ,$
FIELD=OT_HR_PAY     ,ALIAS=                ,USAGE=P5   ,ACTUAL=P2 ,$
FIELD=SSN           ,ALIAS=SSN.IMS         ,USAGE=P9   ,ACTUAL=P3 ,$
FIELD=SSNALPHA      ,ALIAS=SSNALPHA.KEY    ,USAGE=A9   ,ACTUAL=A9 ,$
FIELD=EMPID         ,ALIAS=EMPID.IMS       ,USAGE=A12  ,ACTUAL=A12,$
FIELD=ELAST_NAME    ,ALIAS=ELNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=EFIRST_NAME   ,ALIAS=EFNAME.IMS      ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=RACE          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=ADMIT_DATE    ,ALIAS=ADMDATE.IMS     ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ADMIT_TYPE    ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DISPOSITION   ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=TRANSFER_DATE ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=ALLERGY1      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY2      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY3      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=ALLERGY4      ,ALIAS=                ,USAGE=A15  ,ACTUAL=A15,$
FIELD=HOUSING       ,ALIAS=                ,USAGE=A03  ,ACTUAL=A03,$
FIELD=RPR           ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=URIN          ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PRACTITIONAR  ,ALIAS=                ,USAGE=A02  ,ACTUAL=A02,$
FIELD=SHIFT         ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=PATINET_ID    ,ALIAS=                ,USAGE=P12  ,ACTUAL=P4 ,$
FIELD=WHO_ADDED     ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_ADDED    ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=WHO_EDITED    ,ALIAS=                ,USAGE=A10  ,ACTUAL=A10,$
FIELD=DATE_EDITED   ,ALIAS=                ,USAGE=A08  ,ACTUAL=A08,$
FIELD=STATION_ID    ,ALIAS=                ,USAGE=A12  ,ACTUAL=A12,$
FIELD=DIABETIC      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
FIELD=DIALYSIS      ,ALIAS=                ,USAGE=A01  ,ACTUAL=A01,$
```

Example: **Concatenating Partitioned PCBs**

The following example illustrates how to concatenate partitioned PCBs:

```
PSB=EMPPSBJ, WRITE=NO, PCBNUMBER=4, PL1=NO, $
 SEGNAME=EMPINFO, KEYTYPE=S2, $
PSB=EMPPSBJ, WRITE=NO, PCBNUMBER=5, PL1=NO, $
PSB=EMPPSBJ, WRITE=NO, PCBNUMBER=6, PL1=NO, $
CONCATPCB=4, LOWVALUE=000000001, HIGHVALUE=000001667, $
CONCATPCB=5, LOWVALUE=000001668, HIGHVALUE=000003334, $
CONCATPCB=6, LOWVALUE=000003335, HIGHVALUE=000005000, $
```

Consider the following request. (The key field is named SSNALPHA):

```
SELECT SSNALPHA ...
FROM EMPDBJ
WHERE SSNALPHA = '000001775';
```

The adapter satisfies the request using the EMPDB02 PCB only. If the WHERE clause had requested key values less than 000001775 (rather than equal to 000001775), the adapter would have used the EMPDB01 and EMPDB02 PCBs.

Note:

- If a retrieval request has no WHERE clause, or if the WHERE clause references a non-key field, the adapter accesses all PCBs.
- If any PCB listed in a concatenation lacks the LOWVALUE and HIGHVALUE key range specification, that PCB always participates in the retrieval.

WebFOCUS Reporting With IMS

You can issue any SQL or TABLE request through the Adapter for IMS. However, those requests whose record selection criteria can be applied at the IMS level (with qualified SSAs on .KEY, .IMS, or .HKY fields) result in many fewer DL/I calls and I/O operations than unoptimized requests. They can achieve performance improvements measured in orders of magnitude.

Note: Since IMS does not support the concept of missing data, all fields are considered not missing.

These topics include sample requests with corresponding trace results. You enable traces through Diagnostics on the home pane. In the trace, you get a dump of the SSA buffer setup for each request. For example, the following request contains a selection test on the SEQFIELD search field in the PATINFO segment of the PATDB01 data source:

```
SQL
SELECT LAST_NAME, FIRST_NAME, SEQFIELD, SALARY, SSN
FROM PATDB01
WHERE SEQFIELD = '100000' OR
      SEQFIELD = '100001'
END
```

It produces the following SSA in the trace file:

```
15.13.55 DU D7C1E3C9 D5C6D640 5C604DE2 C5D8C6C9      *PATINFO *-(SEQFI*
15.13.55 DU C5D3C47E 40F1F0F0 F0F0F04E E2C5D8C6      *ELD= 100000+SEQF*
15.13.55 DU C9C5D3C4 7E40F1F0 F0F0F0F1 5D40          *IELD= 100001)   *
```

The output is:

```
LAST_NAME  FIRST_NAME  SEQFIELD  SALARY   SSN

ROSANO     ANDRE       100000    1000.00  197548679
LOVELACE   LARRY       100001    1001.00  197548680
```

Note:

- The dump has a left side and a right side. The left side shows the hexadecimal values in the memory locations. The right side, delimited by *'s, contains the corresponding alphanumeric characters.

  Since not all hexadecimal values represent printable alphanumeric characters, those parts of the dump that correspond to non-printable characters appear as blanks on the right side. In particular, if a comparison value in an SSA is a packed number, it appears as blank on the right-hand side of the dump. In this example, the comparison values are alphanumeric, and therefore they appear.

- The logical operator AND is represented by an asterisk (*) in the SSA, and the logical operator OR is represented by a plus sign (+).

- The field names in the SSA are the alias names from the Master File, since these are the IMS names for the fields.

The SSA always includes the segment name. What distinguishes an optimized request from an unoptimized request is whether the SSA is qualified; a qualified SSA includes the selection test from the request, as in the previous example. An unqualified SSA contains the segment name with no selection criteria.

Whenever possible, the adapter translates screening conditions from the SQL request into qualified SSAs. However, not all server screening conditions can be translated into SSAs. Such conditions are still applied, but by the server, rather than by IMS. In those cases, the adapter retrieves the data sequentially and passes it to the server for screening.

IMS Access Method Restrictions

Some IMS access methods are limited in their ability to apply certain screening conditions. These restrictions can affect request optimization. The following considerations apply to specific access methods:

- Since HDAM data sources are stored randomly and accessed directly, IMS cannot retrieve their records in root key sequence. Therefore, with an HDAM data source, the adapter can only optimize equality conditions on the root key:

  KEY=*value*

  You can build a secondary index on the root key and use it to optimize sequential processing.

  Note: The NE condition is not optimized.

- With HIDAM data sources, the adapter optimizes LIKE tests with a mask by translating the LIKE into a range condition. For an example, see *Partial Key and Multi-Segment Requests in IMS* on page 1067.
- To take advantage of efficiencies available by limiting processing to one area of a Fast Path DEDB data source, the PSB must limit access to that area.

IMS Rules for Constructing SSAs From WHERE Tests

In order for the adapter to translate an SQL WHERE test into a qualified SSA:

- The field tested must be an IMS key field, search field, or a secondary index.
- The test relation must be one of the relations described in these topics.
- The tested field cannot be a floating point numeric or zoned field.

  Note: In order for the adapter to optimize selection tests on zoned key fields, you must describe the USAGE format of these fields as alphanumeric or packed in the Master File.
- The ACTUAL and USAGE formats of the tested field must be basically the same, so no format conversion is required. For example, extending with leading zeros or padding with trailing blanks is not a format conversion, but shifting the implied decimal point in a packed field is a format conversion.
- In an HDAM data source, the tests must use the equality (=) relation on the root key, with no value masking.

The following test relations can be translated to qualified SSAs:

- A comparison of a field to a list of values:

  ```
  fld relation value1 OR fld relation value2 ... OR fld relation valuen
  ```

  For a discussion of SSA generation when comparing a field to a list of values, see *Complex Screening Conditions in IMS* on page 1062.
- = <> < > <= >=
- IN
- NOT IN
- LIKE
- NOT LIKE

- BETWEEN

  The BETWEEN relation is treated as a pair of relations. For example:

  ```
  WHERE field GE . . . AND WHERE field LE . . .
  ```

  A search on a partial key (the high-order portion of a key) is also translated to a range condition and optimized.

- NOT BETWEEN

- Using WebFOCUS, INCLUDES and EXCLUDES on target segments in secondary index databases. For information, see *Qualifying Parent Segments Using INCLUDES and EXCLUDES* on page 1059.

Qualifying Parent Segments Using INCLUDES and EXCLUDES

Using WebFOCUS, you can test whether instances of a given field in a child segment include or exclude all literals in a list using the INCLUDES and EXCLUDES operators. You can also have additional screening conditions for any descendant of the INCLUDES/EXCLUDES parent segment.

Syntax: How to Qualify Parent Segments Using INCLUDES and EXCLUDES

You can connect the literals you are testing for with ANDs and ORs, however, you cannot create compound INCLUDES and EXCLUDES tests with logical AND and OR operators. The Adapter for IMS will always connect all literals used in an INCLUDES/EXCLUDES phrase with the AND condition regardless of the logical predicate used in the request.

A request that contains either of the following phrases returns stock records with cycle number instances for both 00003333 and 00001111:

```
WHERE CYCLNUMB INCLUDES 00003333 OR 00001111
```

or

```
WHERE CYCLNUMB INCLUDES 00003333 AND 00001111
```

For a record to be selected, its cycle number field must have values of both 00003333 and 00001111. If either one is missing, the record will not be selected for the report.

With the following selection criteria, every record that is not associated with the provided secondary index value is selected for the report.

```
WHERE CYCLNUMB EXCLUDES 00003333
```

An EXCLUDE request potentially results in erroneous report output if there is more than one secondary index value associated with the particular target record.

Reference: Usage Notes for INCLUDES and EXCLUDES

- INCLUDES and EXCLUDES are applicable only to the target segments in secondary index databases.
- INCLUDES and EXCLUDES cannot be used when a child segment has type U.
- Literals containing embedded blanks must be enclosed in single quotation marks.
- To use more than one INCLUDES or EXCLUDES phrase in a request, begin each phrase on a separate line.

Example: Using INCLUDES to Qualify Parent Segments

The secondary index database was created based on the IMS IVP sample database Parts.

The Stock Status record was indexed by the CYCLNUMB field in the Cycle Count record, which resulted in the following structural view of the new database:

```
 STRUCTURE OF IMS FILE DIX1DBD ON 12/21/04 AT 13.22.50

STOKSTAT
01 S0
**************
*STOCKEY     **
*STOKFIL1    **
*STOKNUMS    **
*STOKFIL2    **
*            **
***************
 **************
       I
       +-----------------+-----------------+
       I                 I                 I
       I PARTROOT        I CYCCOUNT        I BACKORDR
    02 I U            04 I S0          05 I S0
**************     **************     **************
*PARTKEY     *     *CYCLKEY     **    *BACKKEY     **
*PARTFIL1    *     *CYCLNUMB    **    *FILL1       **
*FILL1       *     *CYCLFIL1    **    *BACKREFR    **
*PARTNAME    *     *CYCLVALU    **    *            **
*            *     *            **    *            **
**************     ***************    ***************
       I             **************     **************
       I
       I
       I
       I STANINFO
    03 I S0
**************
*STANKEY     **
*STANFIL1    **
*STANTYPE    **
*STANFIL2    **
*            **
***************
 **************
```

The following request selects part keys and stock keys for all details that have requested cycle numbers.

```
TABLE FILE DIX1DBD
PRINT CYCLNUMB BY STOCKEY BY PARTKEY BY CYCLKEY
WHERE CYCLNUMB INCLUDES 00003333 AND 00001111
END
```

The output is:

```
STOCKEY     PARTKEY          CYCLKEY    CYCLNUMB
-------     -------          -------    --------
0025906026  02N51P3003F000   20         00003600
                             21         00001111
                             22         00002222
                             23         00003333
0028009126  02RC07GF273J     20         00000190
                             21         00001111
                             23         00003333
```

The report output contains extra CYCCOUNT records that are children of the selected STOCKEY record. To eliminate the extra child records, the request needs an extra screening condition to select only those CYLCNUMB fields that match one of the values in the INCLUDE condition:

```
IF CYCLNUMB EQ 00003333 OR 00001111
```

With this extra condition, the output is:

```
STOCKEY     PARTKEY          CYCLKEY    CYCLNUMB
-------     -------          -------    --------
0025906026  02N51P3003F000   21         00001111
                             23         00003333
0028009126  02RC07GF273J     21         00001111
                             23         00003333
```

Complex Screening Conditions in IMS

The general form for a complex screening condition is:

field1 relation1 value1 OR *field2 relation2 value2* OR ...

If the fields are search or sequence fields, the adapter can optimize the screening test subject to certain conditions. The adapter either constructs a single SSA or multiple SSAs, depending on the characteristics of the segment and the type of relation used.

Note: Specifying the AND operator between logical conditions in a selection test is equivalent to using multiple WHERE statements without the AND. The adapter constructs a qualified SSA that incorporates the AND operation in either case.

The following topics describe:

- The SSA buffer, and how the adapter processes SSAs that are too long to fit into the buffer.
- Conditions under which the adapter constructs a single SSA.
- Conditions under which the adapter constructs multiple SSAs.

Reference: The IMS SSA Buffer

Once the adapter constructs an SSA, it must place the SSA in the SSA buffer in order to submit it in a DL/I call. If the SSA is too long to fit into the buffer, the adapter makes the following choices between the individual screening conditions within the SSA:

- Tests on key fields have priority over all other tests.
- The remaining tests are given priority based on their order in the query statement. Since earlier tests are retained and later tests are eliminated from the SSA, you can control the SSA generation process by carefully constructing your screening conditions.

When an SSA does not fit into the SSA buffer, no error is generated. The selection tests that do not get passed to IMS in the SSA are still applied, but they are not optimized. That is, the server applies them, not IMS.

Note: The number of conditions that fit into the SSA buffer is not fixed. It varies depending on the lengths of the values in the comparisons.

Example: Constructing a Single SSA in IMS

The adapter constructs a single SSA that incorporates the entire complex logical condition if either:

- The segment has no key.
- There is a key, but it is not qualified.
- There is a qualified key, and the relation in the test on the key is the equality (=) relation or a single range.

If the entire SSA fits into the SSA buffer, it is submitted in the DL/I call. If the SSA is too long to fit into the SSA buffer, the adapter makes the choices described in *The IMS SSA Buffer* on page 1063.

The following example illustrates SSA generation when there is no equality test on the key field, but there are tests on a search field:

```
SQL
SELECT LAST_NAME, FIRST_NAME, SEQFIELD, SALARY, SSN
FROM PATDB01
WHERE SEQFIELD IN ('100000','100005') AND
      LAST_NAME IN ('BORRERO','JONES');
END
```

The trace produced is illustrated below. Note that because the sequentially ordered secondary index database is used in the search, the SQL IN condition for SEQFIELD is translated to a range test in the SSA. If the HDAM database itself had been used, the IN condition would have been translated to an equality test in the SSA:

```
14.25.42 DU  set up SSA-Q:
14.25.42 DU D7C1E3C9 D5C6D640 5C604DE2 C5D8C6C9      *PATINFO *-(SEQFI*
14.25.42 DU C5D3C47E 40F1F0F0 F0F0F05C C9E7D5C1      *ELD= 100000*IXNA*
14.25.42 DU D4C54040 6E7EC2D6 D9D9C5D9 D6404040      *ME  >=BORRERO   *
14.25.42 DU 40400000 00000000 00000000 00005CC9      *  ............*I*
14.25.42 DU E7D5C1D4 C540404C 7EC2D6D9 D9C5D9D6      *XNAME  <=BORRERO*
14.25.42 DU 40404040 40FFFFFF FFFFFFFF FFFFFFFF      *     ...........*
14.25.42 DU FF4EE2C5 D8C6C9C5 D3C47E40 F1F0F0F0      *.+SEQFIELD= 1000*
14.25.42 DU F0F05CC9 E7D5C1D4 C540406E 7ED1D6D5      *00*IXNAME  >=JON*
14.25.42 DU C5E24040 40404040 40000000 00000000      *ES       .......*
14.25.42 DU 00000000 005CC9E7 D5C1D4C5 40404C7E      *.....*IXNAME  <=*
14.25.42 DU D1D6D5C5 E2404040 40404040 FFFFFFFF      *JONES       ....*
14.25.42 DU FFFFFFFF FFFFFFFF 4EE2C5D8 C6C9C5D3      *........+SEQFIEL*
14.25.42 DU C47E40F1 F0F0F0F0 F55CC9E7 D5C1D4C5      *D= 100005*IXNAME*
14.25.42 DU 40406E7E C2D6D9D9 C5D9D640 40404040      *  >=BORRERO     *
14.25.42 DU 00000000 00000000 00000000 5CC9E7D5      *............*IXN*
14.25.42 DU C1D4C540 404C7EC2 D6D9D9C5 D9D64040      *AME  <=BORRERO  *
14.25.42 DU 404040FF FFFFFFFF FFFFFFFF FFFFFF4E      *   ............+*
14.25.42 DU E2C5D8C6 C9C5D3C4 7E40F1F0 F0F0F0F5      *SEQFIELD= 100005*
14.25.42 DU 5CC9E7D5 C1D4C540 406E7ED1 D6D5C5E2      **IXNAME  >=JONES*
14.25.42 DU 40404040 40404000 00000000 00000000      *       .........*
14.25.42 DU 0000005C C9E7D5C1 D4C54040 4C7ED1D6      *...*IXNAME  <=JO*
14.25.42 DU D5C5E240 40404040 4040FFFF FFFFFFFF      *NES       ......*
14.25.42 DU FFFFFFFF FFFF5D40                        *......)         *
```

If this SSA did not fit into the SSA buffer, the adapter would retain as much of it as possible in a qualified call, after which the server would apply the remaining tests to the returned segments.

The output is:

```
LAST_NAME  FIRST_NAME  SEQFIELD  SALARY   SSN

JONES      CORNELIUS   100005    1005.00  197548684
```

Syntax: How to Construct Multiple SSAs in IMS

This topic describes how the adapter handles SSA generation when the condition in the SQL request compares a key field to a list of values. The key can be unique or non-unique. The form of such a condition is

```
WHERE key IN (value1, value2, ... ,valuen)
```

where:

key

Is a key field.

value1,...,valuen

Are the comparison values.

In this case, the adapter constructs a separate SSA for each value in the list (in ascending sort sequence) and transmits each one in turn to IMS:

```
(key EQ value1)
  .
  .
  .
(key EQ valuen)
```

The adapter first issues a DL/I call containing only the first SSA. If IMS locates a segment that satisfies the condition in the SSA, the adapter returns the segment to the server. Otherwise, the adapter issues a DL/I call that incorporates only the second SSA. It continues until IMS either locates a segment that satisfies one of the SSAs or exhausts the list of values.

Example: Constructing Multiple SSAs in IMS

In the following example, the adapter constructs three SSAs:

```
SQL
SELECT SSN, SEQFIELD, LAST_NAME
FROM PATDB01
WHERE SSN IN ('197548682', '197548685', '197548691')
END
```

The trace results show the three separate SSAs:

```
14.54.29 DU  set up SSA-Q:
14.54.29 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
14.54.29 DU 4040407E 40F1F9F7 F5F4F8F6 F8F25D40      *   = 197548682) *

14.54.29 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
14.54.29 DU 4040407E 40F1F9F7 F5F4F8F6 F8F55D40      *   = 197548685) *

14.54.29 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
14.54.29 DU 4040407E 40F1F9F7 F5F4F8F6 F9F15D40      *   = 197548691) *
```

The output is:

```
SSN        SEQFIELD  LAST_NAME

197548682  100003    SALEH
197548685  100006    JACA
197548691  100012    BOYCE
```

The next request illustrates an equality test on the key field and an additional test on a search field:

```
SQL
SELECT SSN, SEQFIELD, LAST_NAME
FROM PATDB01
WHERE SSN IN ('197548682', '197548685', '197548691') AND SEQFIELD <
'100012';
END
```

The corresponding trace shows the multiple SSAs that are generated:

```
15.14.05 DU  set up SSA-Q:
15.14.05 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
15.14.05 DU 4040407E 40F1F9F7 F5F4F8F6 F8F25CE2      *   = 197548682*S*
15.14.05 DU C5D8C6C9 C5D3C44C 40F1F0F0 F0F1F25D      *EQFIELD< 100012)*
15.14.05 DU 40                                       *                *

15.14.05 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
15.14.05 DU 4040407E 40F1F9F7 F5F4F8F6 F8F55CE2      *   = 197548685*S*
15.14.05 DU C5D8C6C9 C5D3C44C 40F1F0F0 F0F1F25D      *EQFIELD< 100012)*
15.14.05 DU 40                                       *                *

15.14.05 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
15.14.05 DU 4040407E 40F1F9F7 F5F4F8F6 F9F15CE2      *   = 197548691*S*
15.14.05 DU C5D8C6C9 C5D3C44C 40F1F0F0 F0F1F25D      *EQFIELD< 100012)*
```

The adapter constructs three SSAs and applies them one at a time:

```
(SSN EQ 197548682 AND SEQFIELD LT 100012)
(SSN EQ 197548685 AND SEQFIELD LT 100012)
(SSN EQ 197548691 AND SEQFIELD LT 100012)
```

The output is:

```
SSN        SEQFIELD  LAST_NAME

197548682  100003    SALEH
197548685  100006    JACA
```

If the SSA generated by combining the conditions in all the WHERE statements is too long to fit into the SSA buffer, the adapter retains as much of it as possible in a qualified call by applying the rules described in *The IMS SSA Buffer* on page 1063.

Reference: Sequentially Accessed Root Segments in IMS

If the root segment is the target of the SSA generated by a request, and if it has a unique key, the adapter assumes that IMS will use an index or randomizing scheme to locate the segment without an exhaustive search of the root segment chain. Therefore, the adapter retrieves the segment with qualified GET UNIQUE calls.

Even if the assumption that there is an index or randomizing scheme for IMS to use is not valid, as with HSAM data sources, each call starts its search at the first record in the data source. In this case, it is preferable to describe the root segment as having no key (SEGTYPE=S0), and not to describe any field's alias with the KEY suffix. This causes the adapter to issue qualified GET NEXT calls that access the roots sequentially and maintain the current data source position from one call to the next.

Partial Key and Multi-Segment Requests in IMS

The following topics illustrate SSAs for requests that select on a partial key and requests that access values from multiple segments.

Syntax: How to Optimize Selections on a Partial Key in IMS

Record selection on a partial key can be optimized as a range condition using GE and LE unless the data source is an HDAM data source. The partial key must be the high-order (leftmost) portion of the key. To search on a partial key, use a mask as the comparison value in the relation.

```
WHERE field LIKE 'xxx%'
```

where:

xxx

Are any number of characters that constitute the leftmost portion of the key.

%

Is a wildcard character indicating that any string of characters in this position and beyond satisfies the screening criteria.

Example: Optimizing Selections on a Partial Key in IMS

The adapter translates the LIKE condition in the following request to a range using GE and LE:

```
SQL
SELECT SSN, SEQFIELD, LAST_NAME
FROM PATDB01
WHERE SSN LIKE '1975486%';
END
```

The trace results show that the selection test is translated to a range condition:

```
DU ssa 1
10.09.21 DU D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN  *
10.09.21 DU 4040406E 7EF1F9F7 F5F4F8F6 00005CE2      *   >=1975486..*S*
10.09.21 DU E2D54040 4040404C 7EF1F9F7 F5F4F8F6      *SN     <=1975486*
10.09.21 DU FFFF5D40                                 *..)             *
```

To define the range of values, the adapter appends the lowest hexadecimal value (00) and the highest hexadecimal value (FF) to 1975486 (00 and FF are not alphanumeric representations of numbers; therefore they do not print as such on the right side of the dump).

The output is:

```
SSN        SEQFIELD  LAST_NAME

197548679  100000    ROSANO
197548680  100001    LOVELACE
197548681  100002    BORRERO
197548682  100003    SALEH
197548683  100004    SALGADO
197548684  100005    JONES
197548685  100006    JACA
197548686  100007    PENA
197548687  100008    FREEMAN
```

Example: Generating Multi-Segment Requests in IMS

When a request requires access to multiple segments, the adapter constructs one SSA for each segment. If the selection criteria for a particular segment can be optimized, its corresponding SSA is qualified.

The following request includes an equality test on PARTKEY, the root key of the DI21PART data source, but it contains no selection criteria for the STANKEY field in the STANINFO segment:

```
SQL
SELECT PARTKEY, STANKEY
FROM DI21PART
WHERE PARTKEY = '02AN960C10'
END
```

The SSA for the root segment, PARTROOT, is qualified, but the SSA for the STANINFO segment is not qualified:

```
10.53.47 DU  set up SSA-Q:
10.53.47 DU D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2      *PARTROOT*-(PARTK*
10.53.47 DU C5E8407E 40F0F2C1 D5F9F6F0 C3F1F040      *EY = 02AN960C10 *
10.53.47 DU 40404040 40405D40                        *      )         *

10.53.47 DU E2E3C1D5 C9D5C6D6 5C604040               *STANINFO*-      *
```

The output is:

```
PARTKEY     STANKEY

02AN960C10  02
```

Auto Index Selection

When the Access File defines secondary indexes for a data source, the adapter analyzes each request to determine the most efficient entry point into the data source.

The adapter scans each request to determine whether fields used in record selection tests are key fields, secondary indexes, or the high-order (leftmost) parts of either. Depending on the request criteria, the adapter selects the appropriate PCB for the most efficient access to the data.

Example: Using Auto Index Selection

The PATIENT data source has a secondary index named IXADMD on the ADMIT_DATE field.

In the following request, the field referenced in the WHERE condition is the field associated with the secondary index called IXADMD:

```
SQL
SELECT LAST_NAME,SALARY,ADMIT_DATE
  FROM PATINFO
WHERE ADMIT_DATE = '19920925'
END
```

The trace shows that the adapter generates a qualified SSA using the IXADMD index:

```
11.44.55 DU D7C1E3C9 D5C6D640 5C604DC9 E7C1C4D4      *PATINFO *-(IXADM*
11.44.55 DU C440407E 40F1F9F9 F2F0F9F2 F55D40        *D  = 19920925)  *
```

The output is:

```
LAST_NAME  SALARY   ADMIT_DATE

CAPPARELLI 1024.00  19920925
CAPPARELLI 16682.00 19920925
```

Reference: Selection Tests on Multiple Fields or a Field With Multiple Indexes

If a request includes record selection tests on more than one field, or if a field participates in more than one type of index, the adapter uses the following order of precedence in choosing the PCB to use:

1. .KEY field (primary index).

   If a field in a record selection test is both a .KEY field and the high-order portion of a secondary index, the adapter accesses the data source using the primary index on the .KEY field.

However, if the data source is an HDAM database, and if the request includes a range test on a field that is both an .HKY field and a secondary index field, the adapter accesses the data source through the secondary index field.

2. Secondary index field.
3. .IMS field (search field).

Example: Using Auto Index Selection to Generate a Qualified SSA

The following example demonstrates how secondary indexes and the Auto Index Selection feature affect SSA generation.

The following Master File, AIHDAM, describes an HDAM database. Without a secondary index, the adapter would generate an unqualified SSA on the key field. However, a secondary index is described in the Access File:

```
FILE=AIHDAM   ,SUFFIX=IMS   ,$
SEGNAME=LANGUAGE ,SEGTYPE=S0 ,$
 FIELD=EMPLOYEE_ID6  ,ALIAS=EMPL6.HKY   ,I9 ,I4  ,$
 FIELD=LANGUAGE      ,ALIAS=LANG6.IMS   ,A15 ,A15 ,$
```

The adapter can use the Auto Index Selection feature on the following request to generate a qualified SSA for the range test on the key field:

```
SELECT *
FROM AIHDAM
WHERE EMPLOYEE_ID6 BETWEEN 5248 AND 6393
```

```
14.46.43 DU  set up SSA-Q:
14.46.43 DU D3C1D5C7 E4C1C7C5 5C604DC9 E7C5D4D7      *LANGUAGE*-(IXEMP*
14.46.43 DU F640406E 7E000014 805CC9E7 C5D4D7F6      *6  >=....*IXEMP6*
14.46.43 DU 40404C7E 000018F9 5D40                   *  <=...9)       *
```

Maintaining IMS Data Sources

The Adapter for IMS supports SQL update commands to IMS data sources without the use of remote procedures. For example, UPDATE, INSERT, and DELETE will be translated into equivalent IMS DL/1 calls.

General Guidelines for Maintaining IMS Data Sources

The Structured Query Language (SQL) is intended to be used for access to relational tables, while IMS databases are hierarchical structures. The Adapter for IMS is designed to balance both, so that IMS data structures are treated as if they were relational tables. As this is not always possible, certain rules must be followed.

The adapter is designed to work against a single IMS segment in one path of the IMS database as the target of the SQL Data Manipulation Language (DML) statement. SQL set orientated behavior (multiple updates with one SQL statement) is not supported.

The WHERE clause of the SQL statement must contain all the key fields in the path to the target segment. If a segment does not have a key, at least one field from that segment must be specified in the WHERE clause. Any segment referenced in the SQL query must, using the WHERE clause, be a unique occurrence of that segment. For segments with unique keys, this is guaranteed by IMS, but for segments that have non-unique or no keys, additional fields must be specified in the WHERE clause to uniquely identify the segment. If additional fields are included in the WHERE clause for any segment, including uniquely keyed segments, they will be used in conjunction with the key, to qualify the segment.

Syntax: How to Issue SQL INSERT in IMS

```
INSERT INTO mfdname [(field1, field2 ...)] VALUES (value1, value2 ...)
```

General Rules

The following rules apply to SQL INSERT syntax:

- If you do not specify the field name list, the value list must contain values for all fields in all segments in the Master File as long as the Master File only specifies a single path IMS file.
- If the Master File is multi-path, the field name list must be included and only be for a single path.
- If you specify the field name list, the value list only needs to specify, in field name list order, the field names supplied. As a minimum, you must supply all the key fields for all the segments in the path to the target.
- If a segment is not present, default values will be generated for all non-key fields not supplied, and the missing path segment(s) will be inserted along with the target segment.

Unique Keyed Segment

For a segment with a unique key, the keyfield value is used to insert the target segment. If any additional *fieldname=value* pairs are supplied for a segment in the path, they are used to qualify that path segment.

Any fields not supplied for the target default to type.

Non-unique Keyed Segment

- For a segment with a non-unique key in the target segment or in the path to the target, the segment will be inserted, within keyfield range, according to the RULES=(,FIRST) or RULES=(,LAST) setting for the segment. All segment occurrences of this type whether in the path or target, will be checked for uniqueness, within keyfield range, using the values list.
- If a non-unique segment that is a match is found before the end of the keyfield range, the transaction will be rejected.
- If only the keyfield of the target is supplied, this must be the first segment in the keyfield range, otherwise the transaction will be rejected. No positioning calls will be made. Therefore, RULES=(,HERE) will not be honored. IMS will default to RULES=(,FIRST) for the keyfield value in this situation.

Non-keyed Segment

- A segment with no key or target, or a segment in the path, will be inserted according to the RULES=(,FIRST) or RULES=(,LAST) setting. All segment occurrences of this type, whether in the path or target, will be checked for uniqueness using the values list.
- If a segment is found not to be unique, that is no match is found before the end of the twin chain, the transaction will be rejected. No positioning calls will be made. Therefore, RULES=(,HERE) will not be honored. IMS will default to RULES=(,FIRST) in this situation.

Syntax: How to Issue SQL DELETE in IMS

```
DELETE FROM mfdname WHERE fieldname=value [ AND fieldname=value ... ]
```

General Rules

The following rules apply to SQL DELETE syntax:

- As a minimum, you must supply all the key fields for all the segments in the path to the target. Only one target segment will be deleted. IMS will cascade the delete to dependent segments.
- Additional *fieldname=value* pairs may be included for the target segment and any segment in the path that has no key or a non-unique key. It is recommended to use IMS search fields for any additional *fieldname=value* pairs.

Unique Keyed Segments

For a segment with a unique key, the keyfield value is used to delete the target segment. If you supply any additional *fieldname=value* pairs, they will be used to qualify the deletion.

Non-unique Keyed Segments

For a segment with a non-unique key, you must supply the keyfield. If the WHERE condition(s) for this type of segment, in the path or target, do not identify a unique occurrence, the transaction will be rejected.

Non-keyed Segments

For a segment with no key, you must supply at least one *fieldname=value* pair. If the WHERE condition(s) for this type of segment, in the path or target, do not identify a unique occurrence, the transaction will be rejected. The use of any IMS search fields for this type of segment is highly recommended for efficiency.

Syntax: How to Issue SQL UPDATE in IMS

```
UPDATE mfdname  SET fieldname=value [SET fieldname=value...]
WHERE fieldname=value
[AND fieldname=value]
```

General Rules

The following rules apply to SQL UPDATE syntax:

- As a minimum, in the WHERE condition, you must supply all the key fields for all the segments in the path to the target. Only one target segment will be updated.
- You may include additional *fieldname=value* pairs for the target or any segment in the path. When additional *fieldname=value* pairs are present for the target segment, these will be used for change verification protocol processing. If all target *fieldname=value* pairs that are present match the IMS segment field values, the segment will be updated with the SET *fieldname* values.

Unique Keyed Segments

For a segment with a unique key, the keyfield value and any additional *fieldname=value* pairs are used to qualify the target segment for update.

Non-unique Keyed Segments

For a segment with a non-unique key, you must supply the keyfield. If the WHERE condition(s) for this type of segment do not identify a unique occurrence, the transaction will be rejected. If the set values would result in the updated segment matching an existing segment, the transaction will be rejected.

Non-keyed Segments

For a segment with no key, you must supply at least one *fieldname=value* pair. If the WHERE condition(s) for this type of segment do not identify a unique occurrence, the transaction will be rejected. The use of any IMS search fields for this type of segment is highly recommended for efficiency. If the set values would result in the updated segment matching an existing segment, the transaction will be rejected.

Syntax: How to Obtain the Number of Records Affected by an SQL INSERT, UPDATE, or DELETE Command

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

```
*ENGINE* INT SET PASSRECS {ON|OFF}
```

where:

`INT`

Indicates that the PASSRECS setting in this command will be applied globally to all adapters that support SQL INSERT, UPDATE, and DELETE commands.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Reference: Processing Not Supported for Maintaining IMS Data Sources

The following processing is not supported:

- Null data values.

- Embedded SELECT within INSERT.
- Cursor control, for example, WHERE CURRENT OF on UPDATE or DELETE processing.
- Positioning for RULES=(,HERE) on the SEGM statement for INSERT.
- SQL DML statements using IMS secondary indices.

Commit and Rollback Processing in IMS

IMS data sources can participate in transactions with other types of data sources. These transactions can be coordinated automatically by the XA Transaction Management feature or can be controlled by the user application program.

Commit and Rollback Processing Without XA Transaction Management

When the Adapter for IMS is used for any modification action, it automatically issues a COMMIT after each INSERT, UPDATE, or DELETE call. If the IMS segment processing is part of a logical unit of work (LUW) that involves any relational data sources, the relational transactions should not be committed until the single IMS process has completed successfully. No user rollback of the IMS process is supported. If the IMS process fails, a ROLLBACK is issued by the adapter, and the user application must issue a ROLLBACK of the relational transactions that are part of the LUW.

Commit and Rollback Processing With XA Transaction Management

Although IMS data sources are non-XA compliant, an IMS data source can participate in a two-phase commit handled by the Transaction Coordinator component of the XA Transaction Management feature if:

- The IMS data source is the last participant.
- The IMS data source is the only non- XA compliant participant.

For related information, see *XA Transaction Management* on page 2689.

Metadata Considerations for Maintaining IMS Data Sources

No changes to the Master File are required for maintaining IMS data sources.

Syntax: How to Enable Write Access for IMS

To enable Write access for an IMS data source, specify the WRITE=YES attribute in the Access File.

```
PSB=psbname, WRITE={YES|NO}
```

where:

psbname

Is the name of the FOCPSB library member to use. This name must be identical to the name of the actual PSB that IMS will access. If the member does not exist, the following message is generated:

```
(EDA4261) FOCPSB MEMBER NOT FOUND: psbname
```

YES

Allows SQL update for this database.

NO

Does not allow SQL update for this database. This value is the default.

Reference: Sample IMS Master File for Maintenance Examples

The following database and segment values will be used in all examples.

```
FILENAME=IMSUPDDB,SUFFIX=IMS,$
$
$SEGA has a unique KEY
SEGNAME=SEGA,SEGTYPE=S0,$
  FIELDNAME=FA1    ,ALIAS=FA1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FA2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FA3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
$
$SEGB has a unique KEY
SEGNAME=SEGB,SEGTYPE=S0,PARENT=SEGA,$
  FIELDNAME=FB1    ,ALIAS=FB1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FB2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FB3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
$
$SEGC has a unique KEY
SEGNAME=SEGC,SEGTYPE=S0,PARENT=SEGA,$
  FIELDNAME=FC1    ,ALIAS=FC1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FC2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FC3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
$
$SEGD has a non-unique KEY
SEGNAME=SEGD,SEGTYPE=S0,PARENT=SEGC,$
  FIELDNAME=FD1    ,ALIAS=FD1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FD2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FD3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$

SEGNAME=SEGB,SEGTYPE=S0,PARENT=SEGA,$
  FIELDNAME=FB1    ,ALIAS=FB1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FB2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FB3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
$
$SEGC has a unique KEY
```

```
SEGNAME=SEGC,SEGTYPE=S0,PARENT=SEGA,$
  FIELDNAME=FC1    ,ALIAS=FC1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FC2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FC3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
$
$SEGD has a non-unique KEY

SEGNAME=SEGD,SEGTYPE=S0,PARENT=SEGC,$
  FIELDNAME=FD1    ,ALIAS=FD1KEY.KEY    ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FD2    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
  FIELDNAME=FD3    ,ALIAS=              ,FORMAT=A5,ACTUAL=A5,$
```

Example: Inserting a Root Segment in IMS

The IMS database has one record. The following INSERT statement inserts a second root segment:

```
INSERT INTO IMSUPDDB (FA1,FA2,FA3) VALUES('A2','AAAAA','BBBBB')
```

Example: Inserting a Child Segment in IMS

The following INSERT statement inserts a new SEGB segment:

```
INSERT INTO IMSUPDDB (FA1,FB1,FB2,FB3)
VALUES('A2','A2B1','AAAAA','DDDDD')
```

Example: Inserting Three Segments Under a Root Segment in IMS

The following INSERT statements insert three SEGD segments for root A1.

```
INSERT INTO IMSUPDDB (FA1,FC1,FD1,FD2,FD3)
   VALUES('A1','A1C1','C1D1','D1111','D1111')

INSERT INTO IMSUPDDB (FA1,FC1,FD1,FD2,FD3)
   VALUES('A1','A1C1','C1D1','D2222','D2222')

INSERT INTO IMSUPDDB (FA1,FC1,FD1,FD2,FD3)
   VALUES('A1','A1C1','C1D2','D1111','D1111')
```

Example: Inserting Two Segments on a Single Path of the Data Source in IMS

The following INSERT statement inserts a new SEGC as well as a new SEGD. Default values of blanks or zeros will be used for SEGC for any column not in the values list.

```
INSERT INTO IMSUPDDB (FA1,FC1,FD1,FD2,FD3)
   VALUES('A1','A1C2','C2D2','D2111','D2111')
```

Example: Deleting a Root and Dependent Segments in IMS

The following DELETE statement deletes the root segment A2 and all dependents.

```
DELETE FROM IMSUPDDB WHERE FA1 = 'A2'
```

Chapter 39

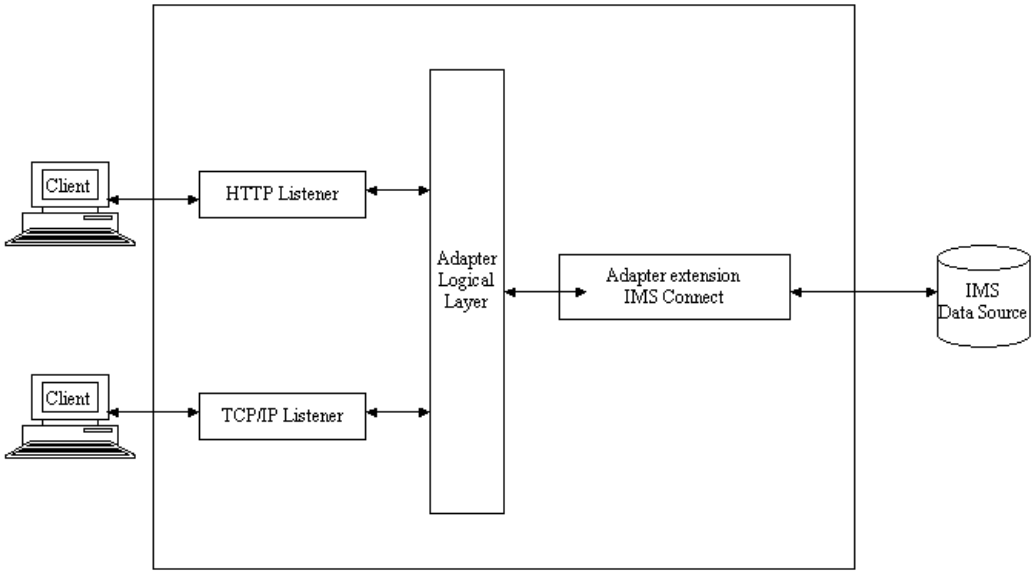
Using the Adapter for IMS Transactions

The Adapter for IMS Transactions processes input parameters, creates and sends requests to the IMS Transactions Server, and creates an answer set based on the received response.

In this chapter:

Preparing the IMS Transactions Environment

The following diagram illustrates the basic flow of the Adapter for IMS Transactions:



This diagram illustrates how the client application communicates with the Application Adapter Server using the TCP/IP or HTTP protocol. The communication to the IMS region uses IMS Connect.

To provide for transparent execution of IMS transactions, metadata describing the program input and output areas is held on the server. This metadata can be derived from COBOL copy book entries, if available.

A Web Console is available as the primary configuration and maintenance tool. It is used to add or change adapter communication parameters and in creating or maintaining the metadata associated with the programs to be executed.

IMS Transactions Adapter Supported Platforms and Release Information

The following platforms are supported:

- Microsoft Windows using IMS Connect/OTMA (any IBM supported version).
- UNIX using IMS Connect/OTMA (any IBM supported version).
- z/OS using OTMA.

Note:

- IBM IMS Connect provides high-performance communications for IMS between one or more TCP/IP or local/390 clients and one or more IMS systems.
- IMS(TM) Open Transaction Manager Access (OTMA) is a transaction-based, connectionless client/server protocol. Though easily generalized, its implementation is specific to IMS in an MVS(TM) sysplex environment. The domain of the protocol is restricted to the domain of the MVS Cross-System Coupling Facility (XCF).
- OTMA addresses the problem of connecting a client to a server so that the client can support a large network, or a large number of sessions, while maintaining high performance.

Configuring the Adapter for IMS Transactions

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to IMS Transactions, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to IMS Transactions takes place when the first query that references that connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

For detailed instructions about declaring connection attributes for your operating system, see *Configuring the Adapter on Windows and UNIX* on page 1082 or *Configuring the Adapter on z/OS* on page 1086.

Configuring the Adapter on Windows and UNIX

The Adapter for IMS Transactions must be configured in order to gain access to IMS programs, create metadata and build the repository. The HWS command *viewers* can provide some of the information required for configuration. The setup information becomes the content of the Adapter for IMS Transactions communications nodes.

To configure the Application Adapter Server for IMS access, perform these steps:

1. Configure the Adapter for TCP/IP on Windows or UNIX.
2. Create Synonyms.

Before you begin this configuration process, you must start the Web Console.

Procedure: How to Configure the Adapter on Windows and UNIX

You can configure the adapter from the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for IMS Transactions on Windows and UNIX

The IMS Transaction adapter is under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

CONNECTION NAME

1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.

HOST

DNS name of the host LPAR of the target IMS region. You can also code the four part IP address.

PORT

Port number on which the IMS Connect is listening. This value may be obtained from viewhws. See *Output From Viewhws* on page 1085.

DATASTORE

ID of the IMS region that provides the transaction processing. This value may be obtained from viewhws. See *Output From Viewhws* on page 1085.

RACF GROUP

RACF Group for the user (when OTMA security is not none.)

SECURITY

There are two methods by which a user can be authenticated when connecting to an IMS Application:

- **Explicit.** If you require all connected users to connect to IMS using the same security profile, select *Explicit* and provide a User ID and Password.
- **Password Passthru.** If Password Passthru is selected, the user ID and password sent from the client application will be used to connect to IMS.

If Explicit security is selected, two additional input fields are shown for User and Password. Enter values that are valid for logging on to IMS.

For more information about available security models, see *Security on Microsoft Windows and UNIX* on page 1085.

User

A user ID that is valid for an IMS logon.

Password

The password associated with the user ID above.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually on Windows and UNIX

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES connection /,: "HOST dns_name
 COM PORT port DATASTORE IMS_region RACFGROUP RACF_Group"
```

where:

`IMSTRAN`

Indicates the adapter.

`connection`

1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.

`dns_name`

DNS name of the host LPAR of the target IMS region. You can also code the four part IP address.

`port`

Port number on which the IMS Connect is listening. This value may be obtained from viewhws.

IMS_region

ID of the IMS region that provides the transaction processing. This value may be obtained from viewhws.

RACF_Group

RACF Group for the user.

For information about security options, see *Security on Microsoft Windows and UNIX* on page 1085.

Example: Declaring Connection Attributes

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES IMS8C /,: "HOST IBIMVS.IBI.COM
 PORT 6686 DATASTORE IMS8C RACFGROUP EDA"
```

Example: Output From Viewhws

```
IEE600I        REPLY TO 01 IS;VIEWHWS
HWSC0001I      HWS ID=HWSB Racf=N
HWSC0001I      Maxsoc=2000 Timeout=8888 23
HWSC0000I      *IMS CONNECT READY* HWSB
HWSC0001I      Datastore=IMS7B Status=ACTIVE
HWSC0001I          Group=IMSGRP7B Memb
HWSC0001I      Datastore=IMS6A Status=ACTIVE
HWSC0001I          Group=IMSGRP6A Member=HWSMEM6A
HWSC0001I          Target Member=TMEM6A
HWSC0001I      Datastore=IMS7A Status=NOT ACTIVE
HWSC0001I          Group=IMSGRP7A Member=HWSMEM7A
HWSC0001I          Target Member=TMEM7A
HWSC0001I          Port=6685 Status=ACTIVE
HWSC0001I          No active Clients
```

Reference: Security on Microsoft Windows and UNIX

If the adapter is deployed on Microsoft Windows or UNIX, starting the server without security is recommended. The client application should provide the User ID and Password to be used for the connection to IMS. This User ID and Password is passed through to IMS for validation. The server does not validate the User ID and Password with the operating system.

If the Explicit option is selected and security information is entered in the adapter configuration window, it overrides this security mode and a single User ID and Password are used for all connected users.

If the adapter is started with security, and Password Passthru is selected, the local security profiles must be synchronized with those of the IMS platform, thereby creating additional security maintenance overhead.

Configuring the Adapter on z/OS

Configuring the adapter consists of specifying connection and authentication information for each connection on z/OS.

Procedure: How to Configure the Adapter on z/OS

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for IMS Transactions on z/OS

The IMS Transaction adapter is under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

CONNECTION NAME

1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.

XCF Member

Target member name. This value may be obtained from /DIS OTMA. See *Output From /DIS OTMA* on page 1088.

XCF Member Group

Target member group name. This value may be obtained from /DIS OTMA. See *Output From /DIS OTMA* on page 1088.

XCF User ID

User ID.

RACF Group

RACF group.

SECURITY

There are two methods by which a user can be authenticated when connecting to an IMS Application:

- **Explicit.** If you require all connected users to connect to IMS using the same security profile, select *Explicit* and provide a User ID and Password.
- **Password Passthru.** If Password Passthru is selected, the user ID and password sent from the client application will be used to connect to IMS.

If Explicit security is selected, two additional input fields are shown for User and Password. Enter values that are valid for logging on to IMS.

For more information about available security models, see *Security on z/OS* on page 1089.

User

Is a user ID that is valid for an IMS logon.

Password

Is the password associated with the user ID above.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Example: Output From /DIS OTMA

```
R 20,/DIS OTMA
IEE600I REPLY TO 20 IS;/DIS OTMA
DFS000I     GROUP/MEMBER      XCF-STATUS   USER-STATUS    SECURITY
IM8C
DFS000I     IMSGRP8C
IM8C
DFS000I     -TMEM8C           ACTIVE       SERVER         NONE
IM8C
DFS000I     -HWSMEM8C         ACTIVE       ACCEPT TRAFFIC
IM8C
```

Syntax: How to Declare Connection Attributes Manually on z/OS

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES connection /,: "XCFMEMB
 target_member_name XCFGROUP target_member_group_name XCFUSMEMB userid
 RACFGROUP racf_group"
```

where:

connection

1-8 character logical name. Select a name that reflects the IMS region you are configuring the connection for.

target_member_name

Name of the target member. This value may be obtained from /DIS OTMA.

target_member_group_name

Name of the target member group. This value may be obtained from /DIS OTMA.

userid

User ID.

racf_group

RACF group.

For information about security options, see *Security on z/OS* on page 1089.

Example: Declaring Connections Attributes on z/OS

```
ENGINE IMSTRAN SET CONNECTION_ATTRIBUTES IMS8C /,: "XCFMEMB TMEM8C
  XCFGROUP IMSGRP8C XCFUSMEMB EDAERH RACFGROUP EDA"
```

Reference: Security on z/OS

If the adapter is deployed on z/OS, you must start the server with security turned on (the default).

The client application must provide a valid User ID and Password that is authenticated by the Adapter for IMS Transactions with calls to security packages (RACF, ACF2 and Top Secret). If the User ID and Password are valid, the subsequent connection to IMS is Trusted.

Managing IMS Transactions Metadata

When the server invokes a transaction or procedure, its needs to know how to build the request, what parameters to pass, and how to format an answer set from the response. For each transaction the server will execute, you must create a synonym that describes the layout of the request/response area.

Creating Synonyms

Synonyms define unique names (or aliases) for each transaction or procedure that is accessible from the server. Synonyms are useful because they hide the underlying transaction or procedure from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows the input parameters and the response layout to be moved while allowing client applications to continue functioning without modification. For example, moving a transaction or procedure from a test region to production. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for IMS Transactions on Windows and UNIX

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Collection of Cobol definitions

Directory

Enter a directory name to display all members in the directory path.

File name/File extension

If you wish to limit retrieval, you can enter a file name and/or file extension:

- In the File name box, enter a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.
- In the File extension box, enter an extension with or without the wildcard symbol %.

Synonym name

Type the name of the synonym.

Program name

Type the name of the IMS Transactions program or APPC transaction.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Input Segment Definition

Is the name of the COBOL FD file that contains information about input parameters.

There can be no more than one set of input COBOL Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

Output Segment Definition

Is the name of the COBOL FD file that contains information about output parameters.

There can be no more than one set of output Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

Reference: Synonym Creation Parameters for IMS Transactions on z/OS

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

File System Selection

Choose one of the following options from the drop-down list:

- *Fully qualified PDS name* to indicate a partitioned dataset on MVS.

  In the input boxes provided, type a PDS name preceded by // and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

  or

- *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

  In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and a File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).

- *Applications* to indicate a pre-configured HFS directory.

Synonym name

Type the name of the synonym.

Program name

Type the name of the IMS Transactions program or APPC transaction.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Input Segment Definition

Is the name of the COBOL FD file that contains information about input parameters.

There can be no more than one set of input COBOL Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

Output Segment Definition

Is the name of the COBOL FD file that contains information about output parameters.

There can be no more than one set of output Copybooks per synonym. You can choose the same Copybook or different Copybooks for input and output parameters.

Example: Master File for IMS TRAN Adapter

```
FILENAME=IMSTPART, SUFFIX=IMSTRAN , $
  SEGMENT=SEG1, SEGTYPE=S0, $
$  GROUP=PARTKEY_INPUT, USAGE=A50, ACTUAL=A50, $
    FIELDNAME=PARTKEY, USAGE=A50, ACTUAL=A50, $
  SEGMENT=SEG11, SEGTYPE=S0, PARENT=SEG1, $
$  GROUP=PARTROOT_OUTPUT, USAGE=A268, ACTUAL=A268, $
    FIELDNAME=FILLER, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=RECTYPE, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=FILLER, USAGE=A22, ACTUAL=A22, $
    FIELDNAME=PARTNUM, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=DESCRIPTION, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=FILLER, USAGE=A26, ACTUAL=A26, $
    FIELDNAME=PROCCODE, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=INVCODE, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FILLER, USAGE=A26, ACTUAL=A26, $
    FIELDNAME=MAKEDEPT, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=PREVNO, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FILLER, USAGE=A26, ACTUAL=A26, $
    FIELDNAME=MAKETIME, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=FILLER, USAGE=A18, ACTUAL=A18, $
    FIELDNAME=COMMCODE, USAGE=A8, ACTUAL=A8, $
```

SEG1, the input segment, was generated from the following COBOL FD:

```
01  PARTKEY-INPUT.
           05 PARTKEY             PIC X(50).
```

SEG11, the output segment, was generated from the following COBOLFD:

```
01  PARTROOT-OUTPUT.
           05 FILLER       PIC X(3).
           05 RECTYPE      PIC X(1).
           05 FILLER       PIC X(22).
           05 PARTNUM      PIC X(12).
           05 FILLER       PIC X(18).
           05 DESCRIPTION  PIC X(20).
           05 FILLER       PIC X(26).
           05 PROCCODE     PIC X(12).
           05 FILLER       PIC X(18).
           05 INVCODE      PIC X(8).
           05 FILLER       PIC X(26).
           05 MAKEDEPT     PIC X(12).
           05 FILLER       PIC X(18).
           05 PREVNO       PIC X(8).
           05 FILLER       PIC X(26).
           05 MAKETIME     PIC X(12).
           05 FILLER       PIC X(18).
           05 COMMCODE     PIC X(8).
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Access File Attributes

Keyword	Description
SEGNAME	Is the name of the input segment in the Master File.
CONNECTION	Indicates the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. Defaults to the default connection.
TRANSACTION	Is the name of transaction to be executed.

Example: Access File for Transaction/Program

```
SEGNAME=SEG1, CONNECTION=IMS8C, TRANSACTION=part, $
```

Note: Do not change the FLOAT or INTEGER parameter values.

Invoking an IMS Transaction

To invoke a IMS Transactions transaction, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, an SQL SELECT statement, or an EX command. (DML is the WebFOCUS internal retrieval language.) For information about the Data Manipulation Language, see the *Stored Procedures Reference* manual.

Syntax: How to Invoke a Transaction Using TABLE

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the IMS Transactions transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

IF/WHERE

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke a Transaction Using SELECT

```
SQL
SELECT [parameter [parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the IMS Transactions transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

WHERE

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Syntax: How to Invoke a Transaction Using EX

```
EX [app_name_space/]syn_name  1=parm1_val,..., N=parmN_val

EX [app_name_space/]syn_name prm1_name=prm1_val,... ,
                             prmN_name=prmN_val

EX [app_name_space/]syn_name [, parm1_val [,...[, parmN_val]]]
```

where:

app_name_space

Is the apps directory name under which the synonyms are stored. This value is optional.

syn_name

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

parmname

Is the name of the parameter taken from the input metadata description.

1=parm1_val

N=parmN_val

prm1_name

prm1_val

parmN_val

Are the parameter values that match the input metadata description.

Example: Invoking the IMSTRAN Transaction

The following requests produce identical output:

```
TABLE FILE IMSTPART
PRINT PARTNUM DESCRIPTION PROCCODE INVCODE MAKEDEPT PREVNO MAKETIME
COMMCODE
IF PARTKEY EQ 'an960c10'
END

SQL
SELECT
PARTNUM,DESCRIPTION,PROCCODE,INVCODE,MAKEDEPT,PREVNO,MAKETIME,COMMCODE
FROM IMSTPART
WHERE PARTKEY='an960c10'
END
```

The output is:

```
PARTNUM  DESCRIPTION PROCCODE INVCODE MAKEDEPT REVNO MAKETIME COMMCODE
-------  ----------- -------- ------- -------- ----- -------- --------
AN960C10 WASHER      74       2       1        2-00  63       14
```

The following EX commands invoke the IMSTRAN transaction:

```
EX IMSTPART 'AN960C10'

EX IMSTPART PARTKEY='AN960C10'

EX IMSTPART 1='AN960C10'
```

The IMSTPART transaction returns the data associated with PARTKEY 'AN960C10':

```
FILE=SQLOUT ,SUFFIX=FIX ,$ SEGNAME=SQLOUT ,$ FIELD=FILLER , E2 ,A3 ,A3 ,
,$ FIELD=RECTYPE , E3 ,A1 ,A1 , ,$ FIELD=FILLER , E4 ,A22 ,A22 , ,$
FIELD=PARTNUM , E5 ,A12 ,A12 , ,$ FIELD=FILLER , E6 ,A18 ,A18 , ,$
FIELD=DESCRIPTION , E7 ,A20 ,A20 , ,$ FIELD=FILLER , E8 ,A26 ,A26 , ,$
FIELD=PROCCODE , E9 ,A12 ,A12 , ,$ FIELD=FILLER , E10 ,A18 ,A18 , ,$
FIELD=INVCODE , E11 ,A8 ,A8 , ,$ FIELD=FILLER , E12 ,A26 ,A26 , ,$
FIELD=MAKEDEPT , E13 ,A12 ,A12 , ,$ FIELD=FILLER , E14 ,A18 ,A18 , ,$
FIELD=PREVNO , E15 ,A8 ,A8 , ,$ FIELD=FILLER , E16 ,A26 ,A26 , ,$
FIELD=MAKETIME , E17 ,A12 ,A12 , ,$ FIELD=FILLER , E18 ,A18 ,A18 , ,$
FIELD=COMMCODE , E19 ,A8 ,A8 , ,$ Part.......... AN960C10;
Desc.......... WASHER Proc Code..... 74; Inv Code..... 2 Make
Dept..... 12-00; Plan Rev Num... Make Time..... 63; Comm Code..... 14
```

Invoking an IMS Stored Procedure

This topic contains overview, configuration, and installation information (as applicable) for the following IMS/TM procedures:

- **CALLIMS LU6.2 for IMS/TM.** This procedure connects to IMS/TM using an IMS/APPC connection from a server for z/OS.
- **CALLITOC OTMA for IMS/TM.** This procedure is available on all servers beginning with Version 5 Release 1.0.

The CALLIMS and CALLITOC procedures for IMS/TM hide the complexity of accessing IMS/TM transactions from the client application. The client application uses any major *standard* access protocol to connect to a server and invoke the adapter to access the IMS/TM transaction. The standard protocols include ODBC, JDBC, OLE DB, or any enabled client or connector.

The Adapter for IMS Transactions allows any client or connector to invoke a transaction running under the control of an IMS/TM transaction-processing monitor. It provides a means of building on existing applications that perform transaction processing against IMS/TM, to create new Web or client/server applications while reusing existing IMS transactions and program logic.

For related information, refer to the *API Reference* manual for API method calls and the *Stored Procedure Reference* manual for details about writing Dialogue Manager procedures.

CALLIMS Procedure for IMS/TM

The CALLIMS procedure is part of the server for z/OS. It is attached to and extended from any compatible API environment, including client applications and Hub Servers.

The CALLIMS procedure uses LU6 communications from a server for z/OS to execute IMS/TM transactions, and passes the output to any client that connects to that server.

CALLITOC OTMA Procedure for IMS/TM

The OTMA procedure fully utilizes the IMS TOC functionality. The IMS TCP/IP Connector (ITOC) and its equivalent product, IMS Connect, is available with IMS Version 7.1 and higher, and allows client TCP/IP communications to and from one or more IMS/TM regions. TOC and IMS Connect are IBM's implementation for allowing the execution of IMS/TM transactions from TCP/IP clients.

Transaction Processing With CALLIMS or CALLITOC

The following steps are performed when a client application calls an IMS/TM transaction using either the CALLIMS or CALLITOC procedure. The difference between the CALLIMS and the CALLITOC procedures is the communications method used to connect to the IMS/TM region from the client application. Also, CALLIMS requires a server for z/OS to initiate the IMS/TM request. Any client on any platform can connect to the server for z/OS.

1. The client application issues the API method call, EDARPC, to run a Dialogue Manager procedure residing on a server (either a Hub Server or a Full-Function Server).

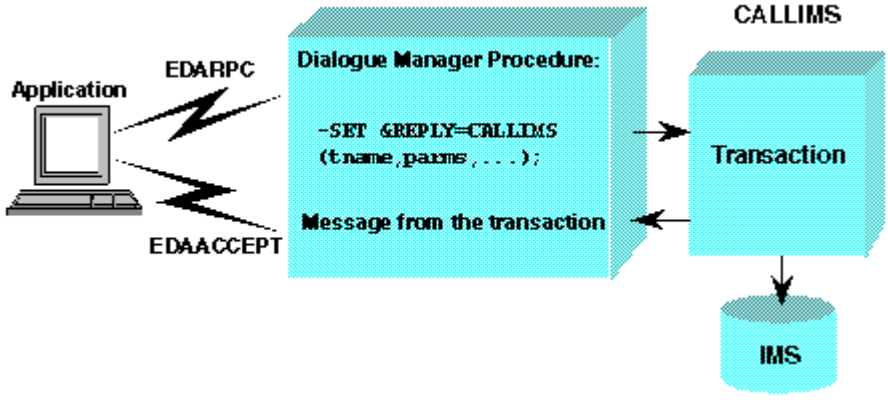
   Parameters are sent as part of the calling sequence for use by the procedure.

2. The Dialogue Manager procedure contains the command -SET, which executes CALLIMS or CALLITOC.
3. CALLIMS or CALLITOC establishes a connection to the IMS/TM environment and invokes the transaction, passing an IMS message as a parameter.
4. The IMS transaction receives and then processes the input message. It then passes one IMS message back to the server. The CALLIMS or CALLITOC procedure returns the message to the client application.
5. The client application issues the method call, EDAACCEPT, to accept the message.

For details on the syntax and use of API method calls, see the *API Reference* manual.

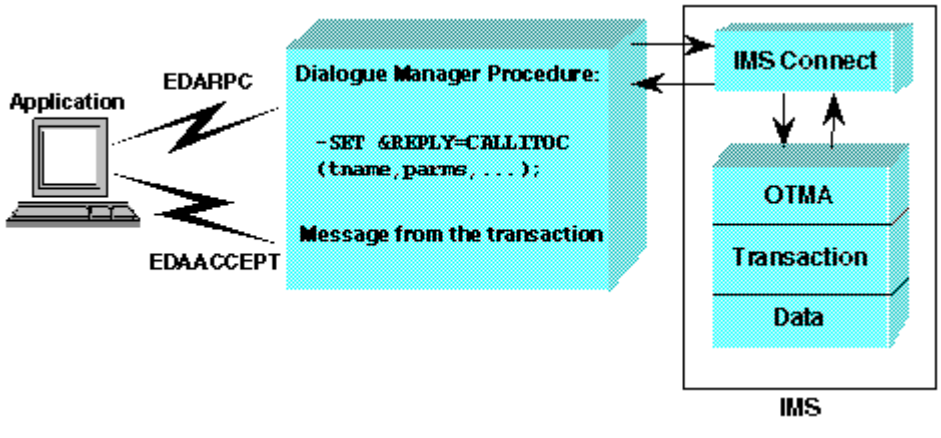
Example: Processing a Transaction With CALLIMS

The following figure illustrates transaction processing as initiated by the client application running a CALLIMS request.



Example: Processing a Transaction With CALLITOC OTMA

The following figure illustrates transaction processing as initiated by the client application running a CALLITOC OTMA request.



Using CALLIMS and CALLITOC

To call an IMS/TM transaction, execute either CALLIMS or CALLITOC from a Dialogue Manager procedure:

- CALLIMS invokes an IMS/TM transaction through the use of APPC/IMS. CALLIMS requires IMS Version 4.1 or higher (IMS/TM) and APPC.
- CALLITOC invokes an IMS/TM transaction through the use of IMS TOC or IMS Connect. CALLITOC requires IMS Version 5.1 or higher (IMS/TM) and a TOC or IMS connection to an IMS OTMA address space.

A sample RPC called CALLIMSC is supplied with the Server. It executes the IMS/TM PART transaction for verification and installation purposes.

Reference: Differences Between CALLIMS and CALLITOC

There are several differences between the CALLIMS and the CALLITOC procedures:

- The communications method used to connect to the IMS/TM region from the client application. CALLIMS uses LU6.2, CALLITOC uses TCP/IP to the mainframe using IMS Connect.
- CALLIMS requires a server for z/OS to initiate the IMS/TM request.
- For CALLIMS, you must run a separate link job to allow CALLIMS to communicate with APPC. For details, see *Installing CALLIMS* on page 1103.

How Data Is Returned With CALLIMS and CALLITOC

CALLIMS and CALLITOC return codes and data to the client application in a series of 72-byte messages. To retrieve the messages, the client application issues the method call EDAACCEPT.

Repeated calls to EDAACCEPT retrieve all messages from CALLIMS and CALLITOC.

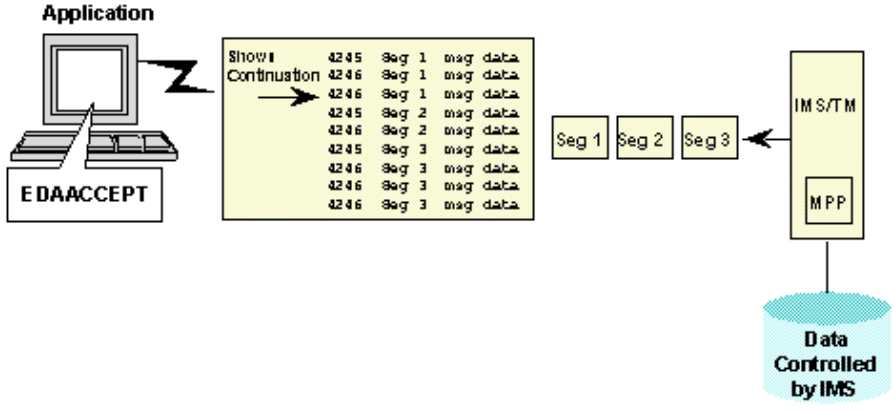
The client application must check the message code in the field scb.msg_code. (The session control block contains several fields pertaining to message processing. For example, scb.msg_code contains a message code and scb.msg_text contains the message text.)

A code of 4245 indicates that the message is coming from CALLIMS or CALLITOC.

For the client application to detect multiple output segments, the Adapter for IMS Transactions returns a message code of 4246 following the 4245, until the end of the segment. At that time, the message code is again 4245, indicating the beginning of a new segment.

Example: **Viewing Message Codes From CALLIMS or CALLITOC**

The following figure illustrates the message codes from CALLIMS or CALLITOC when there are multiple output segments. The client application normally changes the received format to a format acceptable to the end user.



Installing CALLIMS

Complete the following installation steps:

1. *Step 1: Ensure That You Have the Required Hardware and Software*
2. *Step 2: Determine the APPC Setup*
3. *Step 3: Set the Security Level*
4. *Step 4: Link-edit the API*
5. *Step 5: Confirm Interaction With IMS/TM*

Step 1: Ensure That You Have the Required Hardware and Software

CALLIMS requires the following hardware and software:

- ❑ An IBM or IBM-compatible mainframe running z/OS 1.7 or higher. For the latest release information see Information Builders Technical Support web site.
- ❑ APPC Services.
- ❑ An IBM Supported version of IMS.

Step 2: Determine the APPC Setup

CALLIMS communicates with APPC to execute IMS/TM transactions. APPC includes an interface to IMS called APPC/IMS. The APPC/IMS interface must be correctly configured to support IMS/TM transactions.

APPC components that may be used by CALLIMS include:

- **APPLID for the IMS region.** This APPLID identifies applications executing IMS/TM transactions from LU6.2 devices.
- **APPC side information data set.** This data set contains records keyed by a symbolic destination name. This is a client/server feature of APPC that enables a site to establish default information for an application attempting to execute IMS/TM transactions. Each record may contain default APPLIDs, log mode table names, and IMS transaction names.
- **TP profile data set.** This data set enables a site to substitute application-oriented transaction names with a real IMS transaction name. For example, assume that an application refers to a transaction named NAMECHG. The actual transaction (translated in the TP Profile Data set) may be SSNUPDT.

Note: The APPC Transaction Scheduler (ASCH) must be fully configured and running in order for CALLIMS to communicate successfully with APPC/IMS. See the appropriate IBM APPC documentation for configuration instructions.

Step 3: Set the Security Level

To establish conversation security with APPC/IMS, define the environment under which the partner (the other end of the conversation with IMS/TM) will run.

The two possible security levels are:

- **Security Same.** Specifies that the transaction invoked by CALLIMS be run under the same security environment as the calling function (CALLIMS). The security environment includes the user ID and possibly the security profile name of the application.

  This security level is the default if no security information is explicitly provided in the CALLIMS function.
- **Security Program.** Specifies that the application provide a user ID, password, and security profile name to establish the security environment. This security level allows the partner to verify the information, and occurs when the information is provided to the Dialogue Manager procedure in which CALLIMS is invoked.

Step 4: Link-edit the API

The CALLIMS subroutine communicates with APPC using the APPC API. To establish communication between CALLIMS and APPC, you must link-edit the API.

To link-edit the API, submit the following job

qualif.HOME.DATA(GENEAPPC)

where:

qualif

Is the high-level qualifier for the data sets.

Example: Installing CALLIMS

The following JCL is located in member GENEAPPC in the library qualif.HOME.DATA. It must be modified and run.

```
//*******************************************************************
//* NAME:     GENEAPPC JCL
//*
//* FUNCTION: LINKEDIT APPC STUBS INTO CALLIMS
//*
//* PROC SYMBOLIC PARAMETERS:
//*       1. qualif MUST BE CHANGED TO THE HIGH LEVEL QUALIFIER
//*          USED FOR THE PROGRAM DATASETS UNLOAD FROM THE MEDIA
//*
//*******************************************************************

//APPROC  PROC PREFIX='qualif'
//*
//LKED    EXEC PGM=IEWL,PARM='RENT,LIST,NOXREF,LET'
//SYSLIB   DD DISP=SHR,DSN=SYS1.CSSLIB
//SYSIN    DD DISP=SHR,DSN=&PREFIX..HOME.LOAD
//SYSLMOD  DD DISP=SHR,DSN=&PREFIX..HOME.LOAD
//* Switch the above two lines with the following for PDS deployment
//*SYSIN    DD DISP=SHR,DSN=&PREFIX..P.HOME.LOAD
//*SYSLMOD   DD DISP=SHR,DSN=&PREFIX..P.HOME.LOAD
//SYSUT1   DD UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD SYSOUT=A
//*
//APPROC   PEND
//GENFAPPC EXEC APPROC
//LKED.SYSLIN   DD *
  INCLUDE SYSIN(CALLIMS)
  INCLUDE SYSLIB(ATBPBI)
  MODE AMODE(31),RMODE(ANY)
  ENTRY CALLIMS
  NAME CALLIMS(R)
/*
```

Step 5: Confirm Interaction With IMS/TM

The Transaction Server for IMS provides a sample REXX EXEC that helps confirm the successful installation of APPC and APPC/IMS. The EXEC will invoke an existing MPP.

To use the sample EXEC, which is located in *qualif*.HOME.DATA(APPCIVP):

1. You may need to change the default EXEC values to execute the IMS IVP PART transaction.

   The following table lists and describes the default values:

Values	Description
`TP_name = 'PART'`	The name of the transaction to invoke.
`Sym_dest_name = 'IMSDEST'`	The symbolic destination name found in the side information data set.
`Mode_name = 'LU62APPC'`	The log mode table entry name used by the LU6.2 APPLID supplied on the next keyword.
`Partner_LU_Name = 'SCMI41AX'`	The LU6.2 APPLID defined for use by APPC/IMS.
`Partnumber = 'AN960C10'`	The input data for the transaction.

2. In case of errors, APPCIVP calls a standard REXX EXEC named IRXCKRC, so you must copy the EXEC from SYS1.SAMPLIB to *qualif*.HOME.DATA.
3. After you make the necessary changes to APPCIVP and copy IRXCKRC, run the EXEC from TSO:

   ```
   EX 'qualif.HOME.DATA(APPCIVP)' EXEC
   ```

   This routine confirms that APPC/IMS is correctly configured to operate in a client/server environment. The next step enables the server to initiate an IMS connection.

Executing CALLIMS and CALLITOC

CALLIMS or CALLITOC is executed from a Dialogue Manager procedure with the command -SET.

Any of the variables described in the syntax that follows may be implicitly set in the Dialogue Manager procedure using the command -DEFAULTS, or explicitly set on the CALLIMS or CALLITOC call. The variables work the same way, whether set implicitly or explicitly. If both ways are used, the explicit setting takes precedence over the implicit setting. For examples of implicit settings, see *Using -SET to Execute CALLITOC* on page 1111 and *Using -SET to Execute CALLIMS* on page 1110.

Syntax: How to Execute CALLIMS

The dash (-) in the syntax below is used to allow multiple lines for a parameter list in a Dialogue Manager command, as in -SET

```
-SET &REPLY = CALLIMS(&SYM,&TP,&MLEN,&MSG,&DELIM,&OPT,&UID,
- &PW,&SG,&PLU,&LMODE,&HEXCONV,'A1') ;
```

where:

&SYM

Is the symbolic destination name. This is a key value for the side information data set configuration file defined to APPC. APPC supports a symbolic destination name for determining the default APPLID for the IMS/TM region, with a log mode and transaction name. If a value is supplied for &SYM, either implicitly or explicitly, then &TP, &PLU, and &LMODE may be left blank. This field must be 8 characters in length.

&TP

Is the name of the IMS MPP transaction. This field must be 8 characters in length. Set &TP to blanks if you supply a value for &SYM.

&MLEN

Is the length of the IMS MPP message for the transaction. This is set with the Dialogue Manager LENGTH function in a -SET command.

&MSG

Is the input message for the IMS transaction. This message must exactly match the layout expected by the transaction.

&DELIM

Is a non-blank character string used to delimit individual segments in this IMS message. Each delimited segment has its own SEND. This variable is designed for MPP transactions that expect to retrieve multiple segments from the message queue. The length of &DELIM must be 4 characters.

&OPT

Specifies whether CALLIMS waits for a response. Possible values are:

REPL assumes synchronous operation. CALLIMS waits for a return code or other response from IMS/TM.

NORP sends the message, then immediately returns control to the server. CALLIMS does not wait for a response.

The length of &OPT must be 4 characters.

The following parameters are optional. If not used, the parameters must be padded with blanks.

&UID

Is the user ID that invokes the IMS MPP transaction. The length of &UID must be 8 characters.

&PW

Is the password used to invoke the IMS MPP transaction. The length of &PW must be 8 characters.

&SG

Is the security group that the user ID belongs to or is part of. The length of &SG must be 8 characters.

&PLU

Is the LU6.2 APPLID for the IMS/TM region. The length of &PLU must be 8 characters. Set &PLU to blanks if you supply a value for &SYM.

&LMODE

Is the log mode table entry name. The length of &LMODE must be 8 characters. Set &LMODE to blanks if you supply a value for &SYM.

HEXCONV

A value of ON will truncate the message returned after any hex character data. OFF is the default.

'A1'

Sets the format of the response to alphanumeric. It is required by -SET.

Syntax: How to Execute CALLITOC OTMA

The dash (-) in the syntax below is used to allow multiple lines for a parameter list in a Dialogue Manager command, as in -SET

```
    -?SET &REPLY = CALLITOC(&HOST, &PORT, &DSID, &TPNAME, &MLEN, &MESSAGE,
-? &USERID, &RUSERID, &RGROUP, &PASSWD, &DELIM, &HEXCONV, &OPTION, &OTMAEX 'A1');
```

where:

`&HOST`

Is the symbolic destination name. TCP/IP is the host address where IMS Connect or ITOC is running. *&host* is the value in the HOSTNAME=*host* in the HWS configuration file.

`&PORT`

Is the port number that IMS Connect or ITOC is listening on. *&port* is the value in the PORTID=*port* in the HWS configuration file.

`&DSID`

Is the value in the DATASTORE ID=*dsid* in the HWS configuration file.

`&TPNAME`

Is the name of the IMS MPP transaction. This field must be 8 characters in length.

`&MLEN`

Is the length of the IMS MPP message for the transaction. This is set with the Dialogue Manager LENGTH function in a -SET command.

`&MESSAGE`

Is the input message for the IMS transaction. This message must exactly match the layout expected by the transaction.

`&USERID`

Is the ITOC user ID. This user ID is reserved for future use.

`&RUSERID`

Is the user ID that is validated by RACF based on the RGROUP value.

`&RGROUP`

Is the XCF group validated by RACF. *&rgroup* is the value in the GROUP=*rgroup* in the HWS configuration file.

`&PASSWD`

Is the RACF password based on the RUSERID value.

`&DELIM`

Is a non-blank character string used to delimit individual segments in this IMS message. Each delimited segment has its own SEND. This variable is designed for MPP transactions that expect to retrieve multiple segments from the message queue. The length of &DELIM must be 4 characters.

`&HEXCONV`

A value of ON will truncate the message returned after any hex character data. OFF is the default.

`&OPTION`

Specifies whether CALLITOC waits for a response. Possible values are:

`REPL` assumes synchronous operation. CALLITOC waits for a return code or other response from IMS/TM.

`NORP` sends the message, then immediately returns control to the server. CALLITOC does not wait for a response.

The length of &OPT must be 4 characters.

`&OTMAEX`

Specifies which user message exit to use:

- *SAMPLE* indicates HWSSMPL0
- *SAMPL1* indicates HWSSMPL1

The sample user exits enable users to assign their own message formats to fit their business needs. See the IBM documentation for further information about sample user exits.

`'A1'`

Sets the format of the response to alphanumeric. It is required by -SET.

Example: Using -SET to Execute CALLIMS

The following is a sample Dialogue Manager procedure that uses the command -SET to execute CALLIMS. It is member CALLIMS of the data set.

In this example, values for variables are padded with blanks as required. The command -DEFAULTS enables you to specify default values in the Dialogue Manager procedure, but may be overridden by a client application:

```
-DEFAULTS &SYM    = '        '
-DEFAULTS &TP     = 'PART    '
-DEFAULTS &MSG    = 'AN960C10'
-DEFAULTS &DELIM  = '    '
-DEFAULTS &OPT    = 'REPL'
-DEFAULTS &UID    = '        '
-DEFAULTS &PW     = '        '
-DEFAULTS &SG     = '        '
-DEFAULTS &PLU    = 'SCMI41AX'
-DEFAULTS &LMODE  = 'LU62APPC'
-SET &MLEN  = &MSG.LENGTH;
-SET &REPLY =
CALLIMS(&SYM,&TP,&MLEN,&MSG,&DELIM,&OPT,&UID,&PW,&SG,&PLU,&LMODE,'A1');
```

Example: **Using -SET to Execute CALLITOC**

The following is a sample Dialogue Manager procedure that uses the command -SET to execute CALLITOC. For a server for MVS, it is member CALLIMS of the data set.

In this example, values for variables are padded with blanks as required.

```
-DEFAULT   &HOST=    'IBIMVS.IBI.COM ';
-DEFAULT   &PORT=     6683;
-DEFAULT   &DSID=    'IMS61          ';
-DEFAULT   &TPNAME=  'PART           ';
-DEFAULT   &OPTION=  'REPL           ';
-DEFAULT   &MESSAGE= 'AN960C10       ';
-DEFAULT   &USERID=  '               ';
-DEFAULT   &RUSERID= '               ';
-DEFAULT   &RGROUP=  'IMSGRP61       ';
-DEFAULT   &PASSWD=  '               ';
-DEFAULT   &DELIM=   '               ';
-DEFAULT   &HEXCONV= '               ';
-DEFAULTS  &OTMAEX = '*SAMPLE*'

-SET &MLEN = &MESSAGE.LENGTH;

-SET &REPLY = CALLITOC(&HOST,
-                     &PORT,
-                     &DSID,
-                     &TPNAME,
-                     &MLEN,
-                     &MESSAGE,
-                     &USERID,
-                     &RUSERID,
-                     &RGROUP,
-                     &PASSWD,
-                     &DELIM,
-                     &HEXCONV,
-                     &OPTION,
-                     &OTMAEX,
-                     'A1') ;
```

Storing Multiple Messages In a Server File for Later Queries

If you are using a front-end application, such as PowerBuilder, which cannot handle multiple messages, a method is needed to convert the messages into an answer set. This is especially important when using CALLIMS or CALLITOC, which only return messages.

Example: Storing Multiple Messages for Later Execution

The following stored procedure executes a CALLIMS procedure that returns 5 messages. Instead of sending the messages directly to the client, the stored procedure stores the messages in a file. A subsequent SELECT is performed to extract data, as an answer set, from the file. The stored procedure statements required for saving messages in the file are highlighted.

```
-SET &EMGSRV = 'FILE';
DYNAM ALLOC FILE EMGFILE DA qualif.EMGFIL SPACE 2,2 TRACKS UNIT SYSDA -
RECFM FB LRECL 80 BLKSIZE 1600
SET EMGSRV=&EMGSRV
SET PAUSE=OFF
-RUN
-DEFAULT &SYM  = '        '
-DEFAULT &TP   = 'PART    '
-DEFAULT &MSG  = 'AN960C10'
-DEFAULT &DELIM= '        '
-DEFAULT &OPT  = 'REPL    '
-DEFAULT &UID  = '        '
-DEFAULT &PW   = '        '
-DEFAULT &SG   = '        '
-DEFAULT &PLU  = 'SCMI41AX'
-DEFAULT &LMODE= 'LU62APPC'


-SET &MLEN = &MSG.LENGTH ;
-SET &REPLY = CALLIMS(&SYM,&TP,&MLEN,&MSG,&DELIM,&OPT,&UID,
- &PW,&SG,&PLU,&LMODE,'A1') ;


DYNAM FREE DDN EMGFILE
-RUN
DYNAM ALLOC FILE EMGSRV1 DA qualif.EMGFIL SHR REU
-RUN
SQL
SELECT COL1 FROM EMGSRV1;
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
DYNAM FREE DDN EMGSRV1
-RUN
```

For the SELECT COL1 FROM EMGSRV1 statement to work, a Master File has to be predefined for the file. The following is the Master File, EMGSRV1, used for this example:

```
FILENAME=EMGSRV1,SUFFIX=FIX
SEGNAME=ORIGIN,SEGTYPE=S1
 FIELDNAME=COL1,FIRST40,A40,$
 FIELDNAME=COL2,LAST40,A40,$
```

The following is the output, from CALLIMS, which is placed into the file:

```
(FOC4245)  :      Part...........     AN960C10; Desc.......... WASHER
(FOC4246)  :
(FOC4245)  :      Proc Code......           74; Inv Code.......        2
(FOC4245)  :      Make Dept......        12-00; Plan Rev Num...
(FOC4245)  :      Make Time......           63; Comm Code......       14
```

Note: The second line is a FOC4246 message line, which indicates a continuation of the previous line. The sample stored procedure returns the first 40 bytes of all 5 records to the client. The stored procedure can be set to return the entire message.

Chapter 40

Using the Adapter for Informix

The Adapter for Informix allows applications to access Informix data sources. The adapter converts application requests into native Informix statements and returns optimized answer sets to the requesting application. This adapter has read/write capabilities, which enables the adapter to insert the data from an application to the data source.

In this chapter:

Preparing the Informix Environment

The Adapter for Informix minimally requires the installation of the Informix Client SDK software. The Informix Client SDK software allows you to connect to a local or remote Informix database server.

In order to support a greater range of Informix DBMS releases, the current server release uses Informix Client SDK in its Adapter for Informix. All the current Informix Client SDKs support connectivity to wide range of Informix databases. For example, SDK 2.6 will support connectivity to Informix releases 5.1 through 9.3.

Note: Always check the Informix SDK documentation for supported releases of Informix databases.

The following environment settings apply when configuring the server with the Adapter for Informix.

Procedure: How to Set Up the Environment on Windows

On Windows, the Informix environment is set up during the installation of the Informix Server and Informix Client SDK.

If necessary, change the environment variable DBDATE. DBDATE determines how Informix handles conversion between character strings and relational columns of DATE type. If you want to code SQL statements containing date literals in any format other than Y4MD- you must:

1. Ensure that the exported DBDATE value is set to the required date format. Use the Registry Editor (regedit) to search for and clear all registry keys that refer to DBDATE, and then assign the new value to DBDATE using the Informix SDK SetNet program.
2. Specify DBDATE_OVERRIDE OFF in the server profile to ensure that the Adapter for Informix does not override the exported DBDATE value.

   By default, the adapter overrides the values of DBDATE with the value of Y4MD-.

   For more information about DBDATE_OVERRIDE, see *How to Change the Way the Adapter Handles DBDATE* on page 1117.

For example, if you want to code an SQL statement containing a date literal formatted as mm/dd/yyyy (such as 12/31/2006) and execute it using Direct SQL Passthru, you would change the value of DBDATE to 'MDY4/'.

However, it is strongly advised that your applications use the ISO-compatible format Y4MD- for date literals (for example, '2005-12-31'). By doing so, you avoid the need to export DBDATE and to set DBDATE_OVERRIDE.

Procedure: How to Set Up the Environment on UNIX

1. Identify the Informix Client SDK using the UNIX environment variable $INFORMIXDIR. For example, to use SDK 2.9, you would assign its home directory, /rdbms4/isdk29, to INFORMIXDIR:

   ```
   INFORMIXDIR=/rdbms4/isdk29
   export INFORMIXDIR
   ```

2. Identify the server name, as it is specified in the SDK's SQLHOSTS file, using the UNIX environment variable $INFORMIXSERVER:

   ```
   INFORMIXSERVER=server_name
   export INFORMIXSERVER
   ```

3. Specify the path to the Informix shared library using the UNIX environment variable $LD_LIBRARY_PATH. For example:

   ```
   LD_LIBRARY_PATH=$INFORMIXDIR/lib:$LD_LIBRARY_PATH
   export LD_LIBRARY_PATH
   ```

4. If necessary, change the environment variable DBDATE.

   DBDATE determines how Informix handles conversion between character strings and relational columns of DATE type. If you want to code SQL statements containing date literals in any format other than Y4MD- you must:

   a. Ensure that the exported DBDATE value is set to the required date format.

   b. Specify DBDATE_OVERRIDE OFF in the server profile to ensure that the Adapter for Informix does not override the exported DBDATE value.

      By default, the adapter overrides the values of DBDATE with the value of Y4MD-.

      For more information about DBDATE_OVERRIDE, see *How to Change the Way the Adapter Handles DBDATE* on page 1117.

   For example, if you want to code an SQL statement containing a date literal formatted as mm/dd/yyyy (such as 12/31/2006) and execute it using Direct SQL Passthru, you would change the value of DBDATE to 'MDY4/':

   ```
   DBDATE=MDY4/
   export DBDATE
   ```

   However, it is strongly advised that your applications use the ISO-compatible format Y4MD- for date literals (for example, '2005-12-31'). By doing so, you avoid the need to export DBDATE and to set DBDATE_OVERRIDE.

Syntax:

How to Change the Way the Adapter Handles DBDATE

It is recommended that you do not change the way the Adapter for Informix works with the DBDATE environment variable. However, if it is necessary, you can do so by issuing the following command

```
SQL SQLINF SET DBDATE_OVERRIDE {ON|OFF}
```

where:

ON

Sets the DBDATE environment variable to 'Y4MD-' at the moment that the adapter connects to an Informix database.

ON is the default value.

OFF

Ensures that the DBDATE value, at the moment that the adapter connects to an Informix database, is the same value that was in effect immediately after the adapter's environment was set up.

It is recommended that you issue SET DBDATE_OVERRIDE only in the server profile. If it is absolutely necessary to change the value of DBDATE within an individual procedure, one must understand exactly when the new value for DBDATE actually takes effect.

The current value of DBDATE is communicated to the Informix database by the Informix client at the moment that it connects to the database, and remains in effect until the Informix client disconnects from the database. The Adapter for Informix supports multiple concurrent connections. The adapter connects to the Informix database when the adapter starts executing a command that references objects that reside in that database. By default, the adapter does not disconnect until the end of the server agent's session. Setting DBDATE does not affect current connections to the database unless you force the adapter to reconnect. You can force it to reconnect by issuing the following commands:

```
SQL SQLINF SET AUTODISCONNECT ON COMMAND
SQL SQLINF END SESSION
```

Accessing a Remote Informix Server

Using the standard rules for deploying the Informix Client, the server supports connections to:

- Local Informix database servers.
- Remote Informix database servers. To connect to a remote Informix database server, the sqlhosts file on the source machine must contain an entry pointing to the target machine and the listening process must be running on the target machine.

Informix SQL ID (for Informix SE only)

When you access Informix SE, you are identified by your UNIX login ID. The UNIX ID becomes the owner ID for any Informix objects (such as tables or indexes) created with immediate SQL commands. Immediate commands are non-parameterized SQL statements that are passed directly to the data source. The UNIX ID is also used as the sole authorization ID for Informix GRANT and REVOKE statements.

Other types of requests, such as SQL SELECTs, sponsor an Informix search for the necessary privileges to act on the particular tables and views in a data source using the UNIX ID. All Informix security rules are respected.

The following table shows the UNIX privileges required in order to specify certain SQL statements.

Statement	Requires
SELECT	Read permission for the Informix data and index files. UNIX write and execute (search) privileges for the data source directory.
UPDATE	Write permission for the Informix data and index files. UNIX write and execute (search) privileges for the data source directory.

ANSI-Compliant Data Sources

If you create a data source to be ANSI-compliant, owner naming is enforced by Informix. Therefore, to preserve any lowercase owner name, you must enclose it in double quotation marks. Otherwise, Informix will default to uppercase for the owner name. For example:

```
ENGINE SQLINF
SELECT * FROM "user1".test
END
```

For additional information, see the *Informix Guide to SQL*.

XA Support

Read/write applications accessing Informix data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see *XA Support* on page 2689.

Configuring the Adapter for Informix

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to an Informix database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Informix database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Informix Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Informix

The *Informix* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

Informix server name, as defined by the dbservername field in the sqlhosts file.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Name by which the user is known to Informix.

Password

Password associated with the user name.

Database name

Informix database name. For:

- **Informix SE,** specify the entire path to the location of the database files including the database name, but without the dbs extension.
- **Informix Online and Informix Dynamic Server,** specify the database name, but without the dbs extension.

DB_LOCALE

Specifies the database locale, which is what the database server uses to process locale-sensitive data.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication.

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES [connection]
 [datasource]/userid,password ;dbname
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Informix, at connection time, for authentication.

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES [connection]
 [datasource]/;dbname
```

Trusted authentication. The adapter connects to Informix as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES [connection]
 [datasource]/,;dbname
```

where:

SQLINF

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name (or a data source name) used to identify this particular set of attributes.

If you plan to have only one connection to Informix, this parameter is optional. If not specified, the local database server serves as the default connection.

datasource

Is the name of the database server as defined by the dbservername field in the sqlhosts file.

If datasource is not specified, the Informix default server (value of the INFORMIXSERVER environment variable) is used. You can also specify the default server with two consecutive single quotation marks.

userid

Is the primary authorization ID by which you are known to Informix.

password

Is the password associated with the primary authorization ID.

dbname

Also referred to as schema, is the name of the Informix database used for this connection. For:

- **Informix SE,** specify the entire path to the location of the database including the database name, but without the dbs extension. Enclose the entire value in single quotation marks.
- **Informix Online and Informix Dynamic Server,** specify the database name, but without the dbs extension.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allows the application to access the Informix database mydir1/mydir2/sampdb on the Informix database server named SAMPSERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

- For Informix SE:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPSERVER/MYUSER,PASS; 'mydir1/
mydir2/sampdb'
```

- For other versions of Informix:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPSERVER/MYUSER,PASS; SAMPDB
```

The following SET CONNECTION_ATTRIBUTES command connects to the Informix database server named SAMPLESERVER using Password Passthru authentication:

- For Informix SE:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPSERVER/; 'mydir1/mydir2/
sampdb'
```

- For other versions of Informix:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES SAMPSERVER/; SAMPDB
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Informix database server using operating system authentication:

- For Informix SE:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES /,;'mydir1/mydir2/sampdb'
```

- For other versions of Informix:

```
ENGINE SQLINF SET CONNECTION_ATTRIBUTES /,; SAMPDB
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLINF SET DEFAULT_CONNECTION connection
```

where:

`SQLINF`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLINF SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLINF SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLINF

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Informix Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the data source structure and the server mapping of the Informix data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Informix table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for the extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.

- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Informix

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check *Use current database* to use the database that has been set as the default database.
- Select a database from the *Select database* drop-down list, which lists all databases in the current DBMS instance.

  Before selecting a database, if Use current database is checked, uncheck it.

This option does not apply to Informix SE, for which *Use current database* must be checked.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Informix synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLINF ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4,     I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=qatst:"edaqa".nf29004,
CONNECTION=conn_inf,KEYS=1,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Attribute	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Informix table name. The table name can be fully qualified as follows: TABLENAME=[[*database*.]*owner*.]*table*

Attribute	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local Informix database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Attribute	Description
AUTO INCREMENT	Set to Yes to enable autoincrementing.
START	Initial value in incrementing sequence
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Informix Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as variable character (A*n*V).

The following table lists data type mappings based on the value of VARCHAR:

Informix Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
LVARCHAR (*n*)		A2048V	A2048V	A2048	A2048

Informix Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
NVARCHAR (*n*)		A255V	A255V	A255	A255

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE SQLINF SET VARCHAR {ON|OFF}
```

where:

SQLINF

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V). ON is the default value.

OFF

Maps the Informix data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A).

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLINF SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Informix Environment

The Adapter for Informix provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLINF SET PASSRECS {ON|OFF}
```

where:

`SQLINF`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Activating NONBLOCK Mode

The Adapter for Informix has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE SQLINF SET NONBLOCK {0|n}
```

where:

`SQLINF`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Using a Quoted Identifier

The QUOTED_IDENTIFIER setting enables the adapter for Informix to set the Informix environment variable DELIMIDENT, which determines whether the database server interprets strings enclosed in double quotation marks as SQL identifiers or as literal strings.

When the DELIMIDENT environment variable is set to ON, the adapter generates SQL with double quotation marks around identifiers such as table names and column names, when necessary. (For details about the effects of setting the DELIMIDENT environment variable, refer to the Informix documentation.)

The DELIMIDENT environment variable is passed to the Informix database by the Informix client at the moment the client connects to the database; the variable remains in effect until the Informix client disconnects from the database. The adapter connects to the Informix database when the adapter starts executing a command that references objects residing in that database. By default, the adapter does not disconnect until the end of the server agent session. Setting DELIMIDENT does not affect current connections to the database unless you force the adapter to reconnect. You can force it to reconnect by issuing the following commands:

```
SQL SQLINF SET AUTODISCONNECT ON COMMAND
SQL SQLINF END SESSION
```

Tip: You can change this setting manually, or change it from the Web Console by clicking Data Adapters in the menu bar, clicking the name of a configured adapter, and choosing *Change Setting*s from the menu.

Note: The DELIMIDENT variable is supported by the Dynamic, XPS, and SE Informix Servers. However, it is not supported by Informix Online.

Syntax: How to Enable Quoted Identifiers Around Strings

```
ENGINE SQLINF SET QUOTED_IDENTIFIER {OFF|ON}
```

where:

`SQLINF`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`OFF`

Indicates that strings enclosed in double quotation marks are to be interpreted as literal strings. OFF is the default value.

`ON`

Indicates that strings enclosed in double quotation marks are to be interpreted as SQL identifiers.

Informix Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Calling an Informix Stored Procedure Using SQL Passthru

Informix stored procedures are supported using SQL Passthru. These procedures need to be developed within Informix using the CREATE PROCEDURE command.

The adapter supports all scalar parameters: IN, OUT, and INOUT.

Example: Calling a Stored Informix Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLINF syntax.

```
ENGINE SQLINF
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE ON TABLE PCHOLD
END
```

Example: Informix Stored Procedure

```
CREATE PROCEDURE SAMPLE()
RETURNING CHAR(11), CHAR(20), CHAR(15), DATE, CHAR(1);
DEFINE a1 CHAR(11);
DEFINE b1 CHAR(20);
DEFINE c1 CHAR(15);
DEFINE d1 DATE;
DEFINE e1 CHAR(1);
FOREACH entry FOR SELECT ssn5, last_name5, first_name5, birthdate5, sex5
INTO
a1,b1,c1,d1,e1 FROM 'edaqa'.NF29005
RETURN a1,b1,c1,d1,e1 WITH RESUME;
CONTINUE FOREACH;
END FOREACH
END PROCEDURE;
```

The following syntax is used to call the above stored procedure:

```
SQL SQLINF
EX SAMPLE;
TABLE ON TABLE PCHOLD
END
```

Chapter 41

Using the Adapter for Ingres

The Adapter for Ingres allows applications to access Ingres data sources. The adapter converts application requests into Ingres calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Ingres Environment

In order to use the Adapter for Ingres, you must install the Ingres JDBC Driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the Ingres JDBC Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server EDAPRINT log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for Ingres

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to a Ingres data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf), if supported on your platform.

You can declare connections to more than one Ingres data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Ingres

The *Ingres* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Ingres data source. It uses the following syntax:

```
jdbc:ingres://lnxx64r6:21071/qatst;TZ=GMT-5;DATE=INGRESDATE
```

where:

`TZ`

Specifies the Ingres time zone associated with the client's location. It corresponds to the Ingres environment variable II_TIMEZONE_NAME and is assigned the same value. This property is not used directly by the driver, but is sent to the DBMS and affects the processing of dates.

DATE

Specifies the data type of the columns created using the alias keyword *date*. Valid values are: *ansidate* or *ingresdate*. The default value is *ansidate* if the *send_ingres_dates* property is set to false, otherwise it is *ingresdate*. This value is not used directly by the driver, but is sent to the DBMS and affects the processing of query text.

Driver name

Name for the Ingres JDBC driver.

For example: com.ingres.jdbc.IngresDriver

See Ingres documentation for the specific driver release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLING SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLING`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the Ingres data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the Ingres JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMS will encrypt the password before adding it to the server profile.

Note: Consult vendor documentation for the exact name, port, and path.

```
ENGINE SQLING SET CONNECTION_ATTRIBUTES CON1
  'jdbc:ingres://lnxx64r6:21071/qatst;DATE=INGRESDATE;TZ=GMT-5'/uid,pwd
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLING SET DEFAULT_CONNECTION connection
```

where:

`SQLING`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLING SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLING SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLING

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Ingres Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLING to identify the Adapter for Ingres.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLING [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLING

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Ingres

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Ingres Data Type Support* on page 1151.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Ingres synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLING ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Ingres table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Ingres Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Ingres Environment

The Adapter for Ingres provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Ingres.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLING SET TIMEOUT {nn|0}
```

where:

`SQLING`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLING SET PASSRECS {ON|OFF}
```

where:

`SQLING`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Ingres Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 42

Using the Adapter for Interplex

The Adapter for Interplex allows applications to access Interplex data sources. The adapter converts application requests into native Interplex statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Interplex Environment

Prior to configuring the Adapter for Interplex, the following must be installed:

- A middleware product on the Unisys platform.
- An ODBC driver on UNIX or Microsoft Windows/NT.

Configuring the Adapter for Interplex

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to the Interplex database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Interplex database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Interplex Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Interplex

The *Interplex* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLIPX SET CONNECTION_ATTRIBUTES connection
DSN_name/userid,password
```

where:

`SQLIPX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the Interplex Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`userid`

Is the primary authorization ID by which you are known to Interplex.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the Interplex DSN named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLIPX SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console's Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

  ```
  ENGINE SQLIPX SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

  ```
  ENGINE SQLIPX SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
  ```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLIPX SET DEFAULT_CONNECTION connection
```

where:

SQLIPX

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLIPX SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLIPX SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLIPX

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Interplex Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Interplex data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Interplex table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.

- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Interplex

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Interplex synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLIPX ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Interplex table. The value assigned to this attribute can include the name of the owner (also known as schema). and the database link name as follows: TABLENAME=[*owner*.]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Interplex Environment

The Adapter for Interplex provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: **How to Obtain the Number of Rows Updated or Deleted**

```
ENGINE SQLIPX SET PASSRECS {ON|OFF}
```

where:

`SQLIPX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Interplex Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 43

Using the Adapter for iWay Adapter Framework (IWAF)

The Adapter for iWay Adapter Framework allows application to access IWAF enabled data sources.

In this chapter:

Preparing the IWAF Environment

The IWAF adapter has the following requirements:

- **Java:** To use these Java-based adapters, you must have a Java Virtual Machine (JVM) installed and in the path on the server. The JVM can be downloaded from *http://www.java.com.*

  On a Windows server, the JVM location must be added to the PATH. For example:

  ```
  C:\Program Files\Java\jre6\bin\client;C:\Program Files\Java\jre6\bin;
  ```

- **Connectors:** The iWay Connector jar files must be obtained from Information Builders. Please contact Information Builders Technical Support to obtain these files.

- **Adapter files:** These are the jar files for any applications being accessed and are obtained from the application vendor.

  Note: The location of the jar files should be specified in the IWAF_HOME environmental variable.

Procedure: How to Configure the IWAF_HOME Environmental Variable

1. From the Workspace menu, select *Configuration/Monitor*.

   or

   From the Data Management Console, open the *Workspace* and *Configuration/Monitor* folders.

2. On the navigation tree, open the *Configuration Files* and *Miscellaneous* folders.

3. From the *Miscellaneous* folder, right-click *Environment – edaenv.cfg* and select *Edit*.

   The Edit Environment Configuration page opens, showing the location of the file in the window title.

4. Enter the location of the iWay connector and application jar files. For example, if you had placed those files in the c:\ibi\iwaf directory, you would enter:

   ```
   IWAF_HOME=c:\ibi\iwaf
   ```

5. Click the *Save* button to save the file.
6. Stop and restart the server.

Configuring the Adapter for IWAF

Configuring the adapter consist of specifying connection and authentication information for each of the connections you want to establish.

Note: You can declare connection attributes for an adapter that will allow you to create synonyms for either a service or an event.

Procedure: How to Configure the IWAF Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Click *OK* to allow changes to the server configuration.

   The IWAF Successfully added to configuration window appears.

6. Click *Restart Java Services* and then click *OK*.

Procedure: How to Declare Connection Attributes for an IWAF Adapter

You can declare connection attributes for an IWAF adapter from either the Web Console or the Data Management Console.

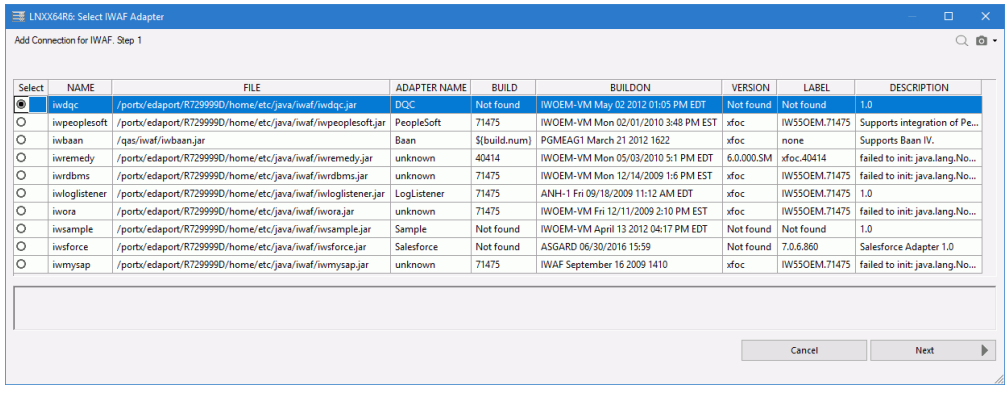
1. From the Web Console menu bar, click *Adapters*

   or

   From the Data Management Console, expand the *Adapters* folder.

   The Adapters folder opens.
2. Expand the *Available* folder, if it is not already expanded.
3. Expand the *Configured* folder, right-click *IWAF*, and select *Add Adapter*.
4. From the list of IWAF adapters, check the radio button for the one you want and click *Next*.

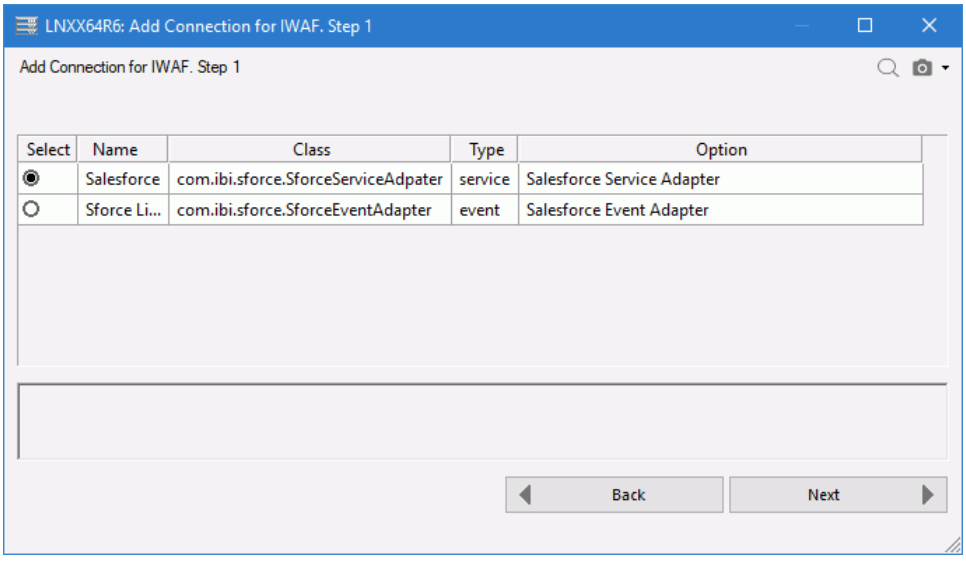
   This list of IWAF adapters will vary from the illustration below depending on what has been installed on your server.



Select	NAME	FILE	ADAPTER NAME	BUILD	BUILDON	VERSION	LABEL	DESCRIPTION
◉	iwdqc	/portx/edaport/R729999D/home/etc/java/iwaf/iwdqc.jar	DQC	Not found	IWOEM-VM May 02 2012 01:05 PM EDT	Not found	Not found	1.0
○	iwpeoplesoft	/portx/edaport/R729999D/home/etc/java/iwaf/iwpeoplesoft.jar	PeopleSoft	71475	IWOEM-VM Mon 02/01/2010 3:48 PM EST	xfoc	IW55OEM.71475	Supports integration of Pe...
○	iwbaan	/qas/iwaf/iwbaan.jar	Baan	${build.num}	PGMEAG1 March 21 2012 1622	xfoc	none	Supports Baan IV.
○	iwremedy	/portx/edaport/R729999D/home/etc/java/iwaf/iwremedy.jar	unknown	40414	IWOEM-VM Mon 05/03/2010 5:1 PM EDT	6.0.000.SM	xfoc.40414	failed to init: java.lang.No...
○	iwrdbms	/portx/edaport/R729999D/home/etc/java/iwaf/iwrdbms.jar	unknown	71475	IWOEM-VM Mon 12/14/2009 1:6 PM EST	xfoc	IW55OEM.71475	failed to init: java.lang.No...
○	iwloglistener	/portx/edaport/R729999D/home/etc/java/iwaf/iwloglistener.jar	LogListener	71475	ANH-1 Fri 09/18/2009 11:12 AM EDT	xfoc	IW55OEM.71475	1.0
○	iwora	/portx/edaport/R729999D/home/etc/java/iwaf/iwora.jar	unknown	71475	IWOEM-VM Fri 12/11/2009 2:10 PM EST	xfoc	IW55OEM.71475	failed to init: java.lang.No...
○	iwsample	/portx/edaport/R729999D/home/etc/java/iwaf/iwsample.jar	Sample	Not found	IWOEM-VM April 13 2012 04:17 PM EDT	Not found	Not found	1.0
○	iwsforce	/portx/edaport/R729999D/home/etc/java/iwaf/iwsforce.jar	Salesforce	Not found	ASGARD 06/30/2016 15:59	Not found	7.0.6.860	Salesforce Adapter 1.0
○	iwmysap	/portx/edaport/R729999D/home/etc/java/iwaf/iwmysap.jar	unknown	71475	IWAF September 16 2009 1410	xfoc	IW55OEM.71475	failed to init: java.lang.No...

   For an explanation of the information in the adapters list, see *IWAF Adapter Information* on page 1174.
5. Select the connection type and click *Next*.

The following image shows the dialog box for Salesforce.com with the Salesforce Service Adapter selected (*service* type). The parameters required will depend on whether you select a *service* or *event* adapter.

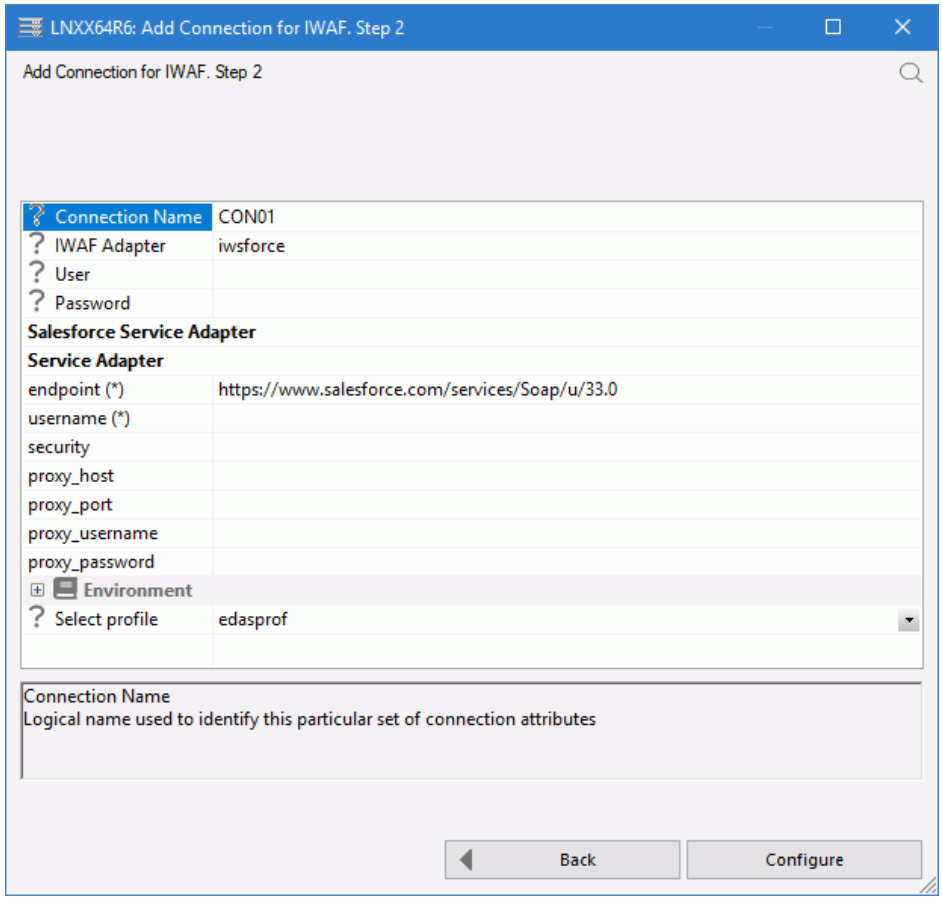


For an explanation of the information in the adapters list, see *IWAF Connection* on page 1175.

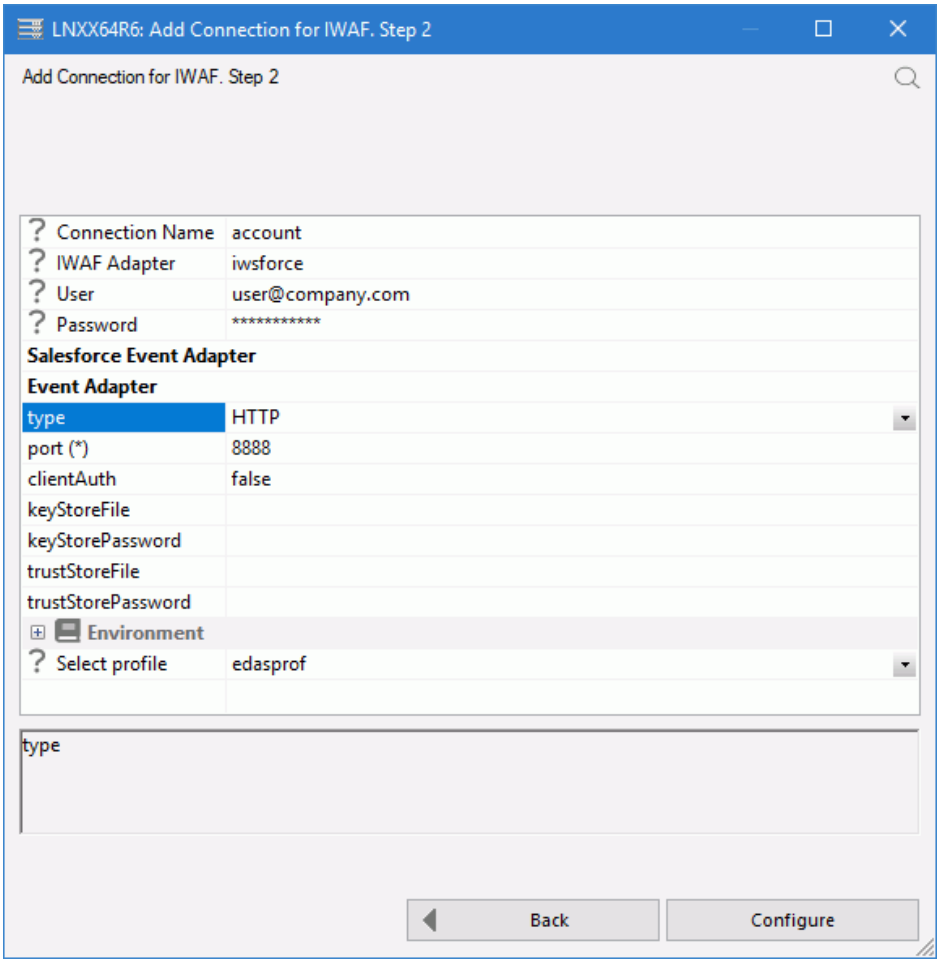
6. Enter values for the parameters required by the adapter and click *Configure*.

   For a description of these parameters, see *IWAF Connection Parameters* on page 1175.

The following illustration shows sample connection parameters for a Salesforce.com service adapter.



The following illustration shows sample connection parameters for a Salesforce.com event adapter.



Reference: IWAF Adapter Information

The following list describes the information in Select IWAF Adapter dialog box.

Name

Internal name of the Adapter.

File

Full path and location of the jar file for this adapter.

Adapter Name

Name of the adapter as it will appear in the Metadata tree.

Build

iWay build number.

Build on

Date and time the adapter was built.

Version

iWay internal build number.

Label

iWay Subversion number.

Description

Adapter description.

Reference: IWAF Connection

The following list describes the information in the Select IWAF Adapter dialog box.

Name

Name of the IWAF adapter.

Class

Name of the class file.

Type

Name of the Java class.

Option

Description of the adapter type.

Reference: IWAF Connection Parameters

The following list describes the information in the Add Connection to IWAF dialog box. Not all options appear for all applications.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

iWAF Adapter

Name of the IWAF adapter.

username (UserName)

User name for this connection.

security (Security Token)

The value provided by salesforce.com.

type (Listener Type)

HTTP or HTTPS.

User

Authorization ID to connect to the source.

Port (listener port)

The port number that will listen for SOAP messages from the application.

Password

Password associated with the authorization ID.

Adapter Specific

Additional options depending on the adapter selected.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Creating Synonyms With iWay Adapter Framework (IWAF)

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note: If you are creating a synonym for a SQLBase data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym for a Service

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Procedure: How to Create a Synonym for an Event

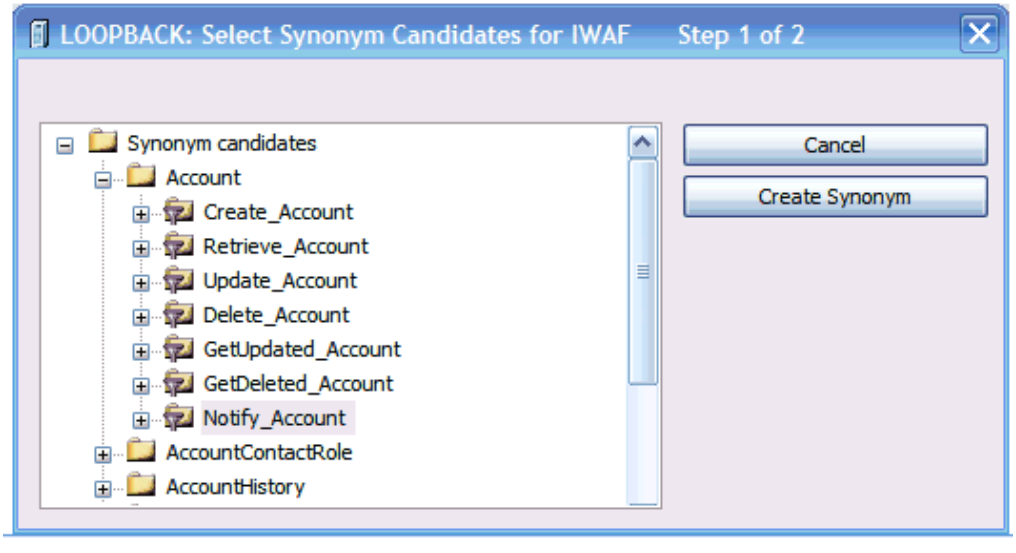
To create a synonym for an event, you must have previously configured the adapter for both the general service and a specific event. You can create a synonym from the Applications or Adapters pages of the Web Console.

1. From the Web Console *Applications* page, click *Get Data*.
2. Click the *New* button and select *Synonym* from the drop-down menu.

   The Select adapter to configure or Select connection to create synonym pane opens.
3. Click a connection to the configured adapter.

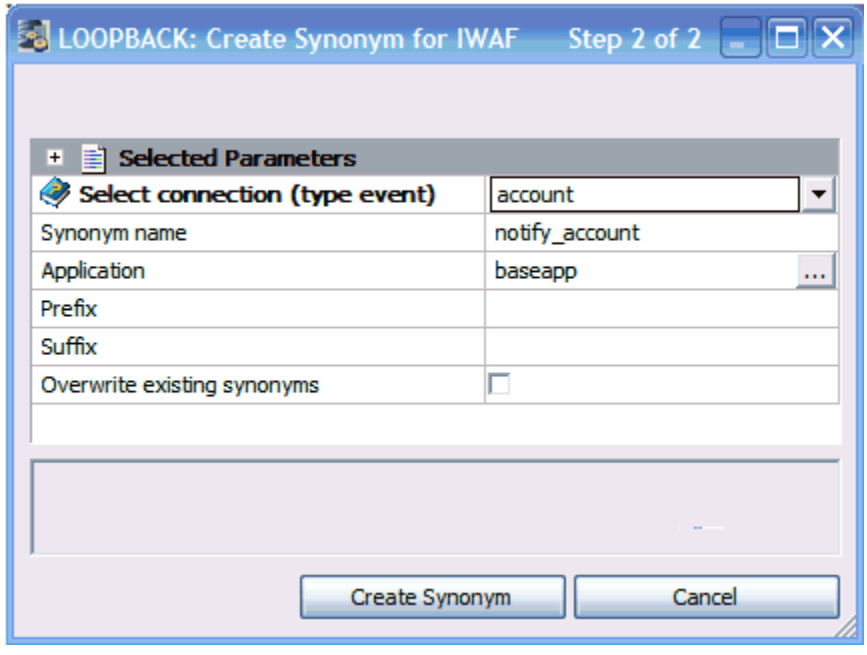
   Note: Even though you are creating a synonym for an event, you must select the associated service to create a synonym.

   The select synonym candidate for IWAF screen opens.
4. Expand a folder, select an event, and click *Create Synonym*. The following image shows the selection of *Notify_Account* from the *Account* folder.



The Create synonym for IWAF screen opens.

5. From the *Select connection (type event)* drop-down menu, select the connection for the event, in this example account.



6. Click *Create Synonym*.

The Status pane indicates that the synonym was created successfully.

Chapter 44 Using the Adapter for JBoss Application Server

The Adapter for JBoss Application Server allows applications to access JBoss data sources. The adapter converts application requests into JBoss calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the JBoss Application Server Environment

In order to use the Adapter for JBoss Application Server, you must install the JBoss driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the JBoss Teiid JDBC driver JAR files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for Teiid JDBC Driver files:

   ```
   CLASSPATH=/usr/jboss/teiid-client.jar:/usr/jboss/teiid-hibernate
   -dialect-7.4.1.jar:$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=C:\jboss-soa-p-5\*\teiid-client.jar;C:\jboss-soa-p-5\*\teiid-
   hibernate-dialect-7.4.1.jar;%CLASSPATH%
   ```

   See JBoss Application Server documentation for specific file names and their locations.

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server's EDAPRINT. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for JBoss Application Server

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to a JBoss data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one JBoss data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for JBoss Application Server

The JBoss Application Server adapter in under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the JBoss data source.

Driver name

Name for the JBoss Teiid JDBC driver.

For example: org.teiid.jdbc.TeiidDrive

See JBoss Application Server documentation for the specific driver release you are using.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLMMX SET CONNECTION_ATTRIBUTES connection 'URL'/userid,password
```

where:

`SQLMMX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the JBoss data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the JBoss Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

Note: Consult vendor documentation for the exact name, port, and path.

```
ENGINE SQLMMX SET CONNECTION_ATTRIBUTES CON1
  'jdbc:jboss:Tmix_Vdb@mm://hostname:31000'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLMMX SET DEFAULT_CONNECTION connection
```

where:

SQLMMX

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLMMX SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLMMX SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLMMX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing JBoss Application Server Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLMMX to identify the Adapter for JBoss Application Server.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLMMX [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLMMX

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using the JBoss Application Server, you must locally install the RDBMS JBoss Driver.

The Server can access third-party database tables across the JBoss Application Server network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note: If you are creating a synonym for a JBoss data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for JBoss Application Server

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 1194.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for JBoss Application Server synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLMMX ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the JBoss table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the JBoss Application Server Environment

The Adapter for JBoss Application Server provides several parameters for customizing the environment and optimizing performance.

Specifying a Payload to Pass to JBoss Application Server

PAYLOAD specifies a JBoss Application Server trusted payload as one or more comma-delimited pairs (represented as '*Key*','*Value*') that you can pass to the JBoss Application Server.

Syntax: How to Specify a Payload to Pass to the JBoss Application Server

```
ENGINE SQLMMX SET PAYLOAD
'KEY1','VALUE1','KEY2','VALUE2',...,'KEYn','VALUEn'
```

Note: Payload can be any 'KEY','VALUE' pair that the JBoss Application Server accepts. There is no limit to the number of pairs that can be sent.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to the JBoss Application Server.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLMMX SET TIMEOUT {nn|0}
```

where:

`SQLMMX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLMMX SET PASSRECS {ON|OFF}
```

where:

`SQLMMX`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

JBoss Application Server Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 45

Using the Adapter for JDBC

The Adapter for JDBC allows applications to access specific JDBC data sources. The adapter converts application requests into JDBC calls and returns optimized answer sets to the requesting application.

For details about JDBC-supported data sources, see the *Server Release Notes*.

In this chapter:

Preparing the JDBC Environment

In order to use the Adapter for JDBC, you must install the JDBC driver for whichever data source you would like to access, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

For example, if you want to access a UniData database, you must install a JDBC driver for UniData. You can configure only one generic JDBC adapter per server instance.

Note:

- If Information Builders offers an adapter for a specific data source type, you should use that adapter to access that type. The Adapter for JDBC does not support data source types for which a specific adapter already exists. For example, Information Builders provides the Adapter for Db2 to access Db2 databases, so you should not use the Adapter for JDBC to access Db2.

- If Information Builders does not currently offer a specific adapter for a data source type, you may use the Adapter for JDBC with a vendor-provided JDBC driver. Note that the adapter conforms to generic JDBC standards. Vendors, on the other hand, may have interpreted or extended those standards for their specific data source requirements. If this introduces an incompatibility, you may experience inconsistent behavior when using the adapter for that vendor's data source type. If this occurs please contact Customer Support Services for assistance. Information Builders will work with your enterprise to try and support your data access needs.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the JDBC Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and checking that the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server EDAPRINT log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for JDBC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to a JDBC data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one JDBC data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for JDBC

The *JDBC* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the JDBC data source.

Driver name

Name for the JDBC driver.

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.

- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLJDBC SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLJDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the JDBC data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLJDBC SET CONNECTION_ATTRIBUTES CON1
  'jdbc:xxxxxxx://hostname:port/datasource
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLJDBC SET DEFAULT_CONNECTION connection
```

where:

`SQLJDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLJDBC SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLJDBC SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLJDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing JDBC Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLJDBC to identify the Adapter for JDBC.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLJDBC [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLJDBC

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using JDBC, you must locally install the RDBMS' JDBC Driver.

The Server can access third-party database tables across the JDBC network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server's global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note: If you are creating a synonym for a JDBC data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for JDBC

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 1211.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for JDBC synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLJDBC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the JDBC table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support Report

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the JDBC Environment

The Adapter for JDBC provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to JDBC.

Syntax: **How to Issue the TIMEOUT Command**

```
ENGINE SQLJDBC SET TIMEOUT {nn|0}
```

where:

`SQLJDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLJDBC SET PASSRECS {ON|OFF}
```

where:

`SQLJDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

JDBC Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 46 Using the Adapter for JD Edwards EnterpriseOne

The Adapter for JD Edwards EnterpriseOne allows WebFOCUS and other applications to access JD Edwards EnterpriseOne data sources. With this adapter, data in the JD Edwards EnterpriseOne DBMS is displayed using rules contained in dictionary files, thereby ensuring that valid information is returned to the requesting program.

This adapter is available for the Windows and IBM i environments. It is intended for use with WebFOCUS Version 7 Release 7 and higher.

In this chapter:

Preparing the JD Edwards EnterpriseOne Environment

Although no environment preparation steps are required, ensure that your system complies with all software specifications.

You must also disable Automatic Passthru to ensure proper processing of JD Edwards EnterpriseOne data.

Reference: Software Requirements

- Any JD Edwards EnterpriseOne Application installation.
- DBMS connectivity, as required by the installed EnterpriseOne environment. The following DBMSs are currently supported:
  - Oracle

- Microsoft SQL Server
- DB2

Note: This adapter can only be configured with reporting servers on the Windows and IBM i operating systems.

Syntax: How to Disable Automatic Passthru

To turn Automatic Passthru off, enter the following command in the server profile, edasprof.prf:

```
SQL
   SET APT = OFF;
END
```

Overview of the Setup Process

The setup process for the Adapter for EnterpriseOne is comprised of the following basic steps:

1. **Create the JD Edwards EnterpriseOne synonyms.** You use the synonym creation facility for Db2, Microsoft SQL Server, or Oracle.

   Note: For IBM i, see *How to Create Synonyms for JD Edwards EnterpriseOne Under IBM i* on page 1219
2. **Configure the Adapter for JD Edwards EnterpriseOne.** This indicates whether you will be using Group-based or Role-based security.
3. **Refresh security extracts.** You will need to perform this step initially, and repeat it only if security information changes.
4. **Refresh the metadata repository.** You will need to perform this step initially, and repeat it only if there are changes in the metadata for tables. This occurs infrequently at most sites.
5. **Convert the synonyms for JD Edwards EnterpriseOne.**
6. **Set UDCDIC parameters (Windows only).** This step is required before you can run reports.

Configuring the Adapter for JD Edwards EnterpriseOne

This process defines connection and authentication information.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Configuration Parameters and Tasks for JD Edwards EnterpriseOne

The *JD Edwards EnterpriseOne* adapter is under the *ERP* group folder.

The following list describes the connection attributes for which you can supply values, and additional tasks you must perform to complete the configuration for this adapter.

Security Authentication

- Check this box if the reporting server is secured. This option applies if every JD Edwards EnterpriseOne user has a user ID on the reporting server system.
- Leave the box unchecked if the reporting server is unsecured. This option applies if every JD Edwards EnterpriseOne user's user ID and password are stored and are being authenticated by Managed Reporting and/or a self-service application:
  - For Managed Reporting, the user ID must be the JD Edwards EnterpriseOne user ID.
  - For Self-Service requests, the JD Edwards EnterpriseOne user ID must be passed to the server.

  In either case, the IBI_REPORT_USER variable must be set to the JD Edwards EnterpriseOne user ID.

ROLE-based security

When you configure the JD Edwards EnterpriseOne adapter, Group-based security is used by default. Leave the box unchecked, if you wish to use Group-based security.

Check this box if you wish to use Role-based security.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

When you click the *Configure*, the adapter is added to the Adapters list in the Web Console or the Adapters folder of the Data Management Console.

Creating Synonyms for JD Edwards EnterpriseOne

Before you can use the Adapter for JD Edwards EnterpriseOne, you must create synonyms for the data you will be reporting against using one of the data adapters that is supported on your platform, and set the paths required to access those synonyms:

- In the Windows environment, you can create synonyms using the Adapters for Db2, Oracle, or Microsoft SQL Server.
- In the IBM i environment, you can create synonyms using Db2.

Procedure: How to Create Synonyms for JD Edwards EnterpriseOne

Note: To create a synonym under IBM i, see *How to Create Synonyms for JD Edwards EnterpriseOne Under IBM i* on page 1219

From the Web Console or the Data Management Console:

1. Configure the adapter you wish to use to create synonyms for the JD Edwards EnterpriseOne data.
2. Use the configured adapter to create synonyms for the JD Edwards EnterpriseOne tables you wish to use. During this process, you will select the tables for which you wish to create synonyms and specify the application in which the tables will be stored.

For details about configuration and synonym creation steps for supported data adapters, see *Using the Adapter for Db2* on page 497, *Using the Adapter for Microsoft SQL Server* on page 1399, or *Using the Adapter for Oracle* on page 1769.

Procedure: How to Create Synonyms for JD Edwards EnterpriseOne Under IBM i

1. From the Web Console *Applications* page, click *Get Data*.
2. Double-click a Db2 connection.

   The first of a series of synonym creation panes opens.
3. Enter values for the parameters required by the adapter as described in *Synonym Creation Parameters for JD Edwards EnterpriseOne Under IBM i* on page 1219.
4. After entering the parameter values, click *Create Synonym*.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Reference: Synonym Creation Parameters for JD Edwards EnterpriseOne Under IBM i

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select Synonym Candidates for Db2 (CON) Step 1

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Library

Type a string for filtering the Library (or Db2 Collection), inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.

Object Name

Type a string for filtering the table, view, or object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables, views, or objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Note: When you create a synonym for Db2 on the IBM i platform, standard IBM i naming conventions apply to the target data source. Therefore, the Adapter for Db2 supports the use of double-quotation marks around any library name and/or file name that contains lowercase or NLS characters.

Create Synonym for Db2 (CON) Step 2 of 3

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

One-part name

On the IBM i platform, the *One-part name* check box is unchecked by default. The unchecked behavior generates a table name that includes the explicit name of the library containing the table. For example, if you specified a library on the first Create Synonym pane, a qualified name like the following is automatically created in the Access File:

```
TABLENAME=MYLIB/MYTABLE
```

With this explicit type of entry in the Access File, at run-time the library is directly located and searched for the table name. If you select the check box, the explicit library name is not stored in the metadata (Access File). When the synonym is generated, the library portion of the table name is omitted from the Access File, and appears as follows:

```
TABLENAME=MYTABLE
```

With this type of entry in the Access File, at run time the library path of the user is searched until the table name is located.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Create Synonym for Db2 Step 3 of 3

Select date format

The options are: YMD, YYMD, DMY, MDY, MDYY, DMYY, MYY,YYM. (YYMD is the default setting.) The selected format will be used only if the field is described as a DATE in the DataDictionary.

Presumptive Join

Select the *Presumptive Joins* check box to include additional DEFINEs (virtual fields) for presumptive join fields in the synonym.

The default setting is *ON*.

Field Names

Select the *Long Fieldname* radio button (the default) to display the field descriptions as names on reports. Select the *Short Fieldname* radio button to use the JDE aliases as field names on reports. The default setting is *Long Fieldname*.

Language Code

Enter the appropriate Language Code, which exists in the JDE F9292 file. (Leave the field blank for English.)

UDC

Select the *UDC* check box to ensure that UDC description fields are generated as DEFINEs (virtual fields) in the synonym. The default setting is *ON*.

Combine UDC

Select the *Combine UDC* check box to Combine User Defined Code. The default setting is *OFF*.

Reference: Ensuring Master File Compatibility Between IBM i and Windows

Before you begin the configuration procedure, it is advisable to consider whether you have procedures that were designed to run on the IBM i platform, which you may wish to run in a Windows environment. If you anticipate this need, your first task is to synchronize the Master Files generated on Windows with those generated on IBM i using the amper variable &MFDCOMP in the jdegdinc.fex file.

The default variable setting is:

```
-DEFAULT &MFDCOMP='N';
```

To ensure matching field names on IBM i and Windows, revise the setting to:

```
-DEFAULT &MFDCOMP='Y';
```

Refreshing the Metadata Repository

The Metadata repository contains the dictionary information for the JD Edwards EnterpriseOne tables.

You must refresh the repository the first time you set up the adapter and repeat the process each time the JD Edwards EnterpriseOne tables change. (At most sites, changes occur infrequently.)

Important: This step must be done before you convert any Master Files (synonyms).

Procedure: How to Refresh the Metadata Repository on Windows

From the Adapters list in the navigation pane on the Web Console or the Adapters folder of the Data Management Console:

1. Right-click the configured JD Edwards EnterpriseOne adapter.
2. Choose *Refresh Metadata Repository* from the menu. The Refresh Metadata Repository pane opens. The JDE tables required for this procedure are listed in the first column.
3. For *each* table, select the suffix of a previously configured DBMS from the drop-down list.

   Note: The listed tables may be associated with different DBMSs.
4. In the Qualifier input boxes enter the qualifier associated with each table, using the syntax required for the selected DBMS:
   - For SQLMSS: `[database.]owner`
   - For Db2: `[location.]owner`
   - For SQLORA: `owner`
5. In the Connection input boxes, enter the same connection name you used when configuring the selected DBMS.
6. Click *Refresh Now*. The refresh occurs transparently. A message indicates the process was successful.

Procedure: How to Refresh the Metadata Repository on IBM i

From the Adapters list in the navigation pane on the Web Console or the Adapters folder of the Data Management Console:

1. Right-click the configured JD Edwards EnterpriseOne adapter and select *Refresh Metadata Repository*.

You will need to perform this step initially, and repeat it only if there are changes in the metadata for tables. This occurs infrequently at most sites.

The Refresh Metadata Repository pane opens. The JDE tables required for this procedure are listed in the first column.

2. For *each* table, select the suffix of a previously configured DBMS from the drop-down list.

   Note: The listed tables may be associated with different DBMSs.

3. Click *Refresh Now*.

Refresh Security Extracts

This process creates security extract files that are stored as FOCUS files in the JOWSEC application that is created when you configure the adapter.

You must repeat this step whenever security extract information needs to be updated.

Procedure: How to Refresh Security Extracts on Windows

From the Adapters list in the navigation pane or the Adapters folder of the DMC:

1. Right-click the configured JD Edwards EnterpriseOne adapter.
2. Select *Refresh Security Extracts*.

   The Refresh Security Extracts pane opens, indicating the Security mode you selected during adapter configuration: ROLE-based or GROUP-based (see *Configuration Parameters and Tasks for JD Edwards EnterpriseOne* on page 1217). Note that the required JDE tables that are listed in the first column vary depending on the Security mode that is in effect.

3. For each table, select the suffix of a previously configured DBMS from the drop-down list.

   Note: The listed tables may be associated with different DBMSs.

4. In the Qualifier input boxes enter the qualifier associated with each table, using the syntax required for the selected DBMS:

   - For SQLMSS: `[database.]owner`
   - For Db2: `[location.]owner`
   - For SQLORA: `owner`

5. In the Connection input boxes, enter the same connection name you used when configuring the selected DBMS.
6. Click *Submit*.

   The refresh occurs transparently. A message indicates the process was successful.

Procedure: How to Refresh Security Extracts on IBM i

From the Adapters list in the navigation pane or the Adapters folder of the DMC:

1. Right-click the configured JD Edwards EnterpriseOne adapter.
2. Select *Refresh Security Extracts*.

   The Refresh Security Extracts pane opens, indicating the Security mode you selected during adapter configuration: ROLE-based or GROUP-based (see *Configuration Parameters and Tasks for JD Edwards EnterpriseOne* on page 1217).

   The JDE tables required for this procedure are listed. Note that the list varies depending on the Security mode that is in effect.
3. In the input boxes provided, enter a library name for each of the JDE tables.
4. Click *Submit*.

   The refresh occurs transparently. A message indicates that the process has been successful.

Converting Synonyms for JD Edwards EnterpriseOne (Non IBM i Platforms Only)

You are now ready to convert synonyms. The conversions of the synonyms are affected by your option entries on the Master File Conversion pane. The following entries are added to the Master File:

- UDC (user-defined code) descriptor fields (if you check UDC on the Master File Conversion pane).
- Date fields are redefined as Gregorian dates, with the data format you select on the Conversion pane.
- Numeric fields are redefined with the correct decimal precision, as defined in the data dictionary repository.

Procedure: How to Convert Synonyms

1. From the Adapters list in the navigation pane, right-click *JD Edwards EnterpriseOne*, and select *Convert Master Files*. The Master Files Conversion pane opens at Part 1 of 2.
2. From the drop-down list, select the application in which you created the synonyms, then click *Next*.
3. On the second Master Files Conversion pane, select the option you wish to use.

The following image shows the second part of the Master Files Conversion pane.

- Select a date format from the drop-down list. The options are: YMD, YYMD, DMY, MDY, MDYY, DMYY, MYY, YYM. (YYMD is the default setting.) The selected format will be used only if the field is described as a DATE in the Data Dictionary.
- Select *N* for *Redefine*. IBM i only: *Y* is used for compatibility with earlier releases.
- Check the *UDC* box to ensure that UDC description fields are generated as DEFINEs (virtual fields) in the synonym.
- Do not check the Combine UDC box. IBM i only: It is used only for compatibility with earlier releases.
- Leave the *Hybrid* box unchecked (the default). (Windows only.)
- Under Options, do one of the following:

  Click the *Select Master File from a list* radio button, then from the drop-down list either select a file or choose *All*.

  or

  Click the *Enter specific filename to process* radio button and type a file name in the box provided.

4. Click *Convert Master File(s)* to start the conversion.

   A message is displayed when the process is successfully completed.

   Tip: You can click the *Back* button if you wish to return to the first synonym Conversion pane.

Except on Windows, you are now ready to report against these tables. On Windows, proceed to *Setting the UDCDIC Environment Variable (Windows only)* on page 1226.

Setting the UDCDIC Environment Variable (Windows only)

On Windows, before you can run reports against your synonyms, you must set the UDCDIC environment variable to locate a file that stores codes which are mapped to field names. This mapping provides significant data input benefits. For information about this variable, see *How to Set the UDCDIC Environment Variable on Windows* on page 1226.

Syntax: How to Set the UDCDIC Environment Variable on Windows

This step is required only on the *Windows* platform to ensure that the server can locate the udcdic.ftm file, which stores codes that are mapped to field names. This mapping provide significant data display benefits.

You can set the UDCDIC environment variable in a Managed Reporting user profile, outside of Managed Reporting as a variable in a procedure, or as a Windows environment variable where the server is installed.

Method 1: To set the variable in a Managed Reporting user profile or a procedure, use the following syntax

```
SETENV UDCDIC=reference_dir
```

where:

reference_dir

Is the path to the reference directory that contains the appropriate udcdic.ftm file.

For example,

```
SETENV UDCDIC=C:\ibi\apps\jowsec
```

This method is more flexible than method 2 because you can quickly change the path designation to point to a different udcdic.ftm file. (For example, this technique would be useful if you need to report against multiple UDC environments).

Method 2: To set UDCDIC as a Windows environment variable, navigate to the Windows facility for defining environment variables and enter:

Variable name: *UDCDIC*

Variable value: the path where the udcdic.ftm file is located. For example,

```
\ibi\apps\jowsec
```

The location of the reference directory can be anywhere on the drive. The UDCDIC environment variable is used to point to that location.

Chapter 47

Using the Adapter for JD Edwards World

The Adapter for JD Edwards World allows WebFOCUS and other applications to access JD Edwards World data sources. With this adapter, data in the JD Edwards World DBMS is displayed using rules contained in dictionary files, thereby ensuring that valid information is returned to the requesting program.

This adapter is available on the IBM i platform (formerly known as i5/OS).

In this chapter:

Installation Prerequisites

The software requirements for the Adapter for JD Edwards World are:

- **Operating System:** V5R4M0 or higher.
- Server Version 7 Release 7 or higher, should be installed and working. For details, see the *Server Installation* manual.

  Note: If the standard IADMIN ID was not used to install the Server, manually change ownership to the ID that was used.
- RDBMS connectivity - A Db2 connector must be configured. For more information, see *Using the Adapter for Db2* on page 497.

You must also verify that your end-user program can access the Server.

Configuring the Adapter for JD Edwards World

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

After the adapter is configured, you must refresh the metadata before you can create synonyms.

Procedure: How to Configure the Adapter for JD Edwards World From the Web Console

To configure the adapter:

1. From the Web Console Applications page, click *Get Data*.
2. Find the adapter on the Available list in the Web Console.
   You can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. Expand the *JD Edwards World* folder, and double-click the *A7.x - A9.x* connection.

   The Add JD Edwards World to Configuration pane opens.
4. Leave Security options unchecked (OFF) at this point. You will add them later.
5. Select a profile from the drop-down list to indicate the level of profile that the server is running: global, service, group, user. The standard server global profile, edasprof.prf, is the default.
6. Click *Configure* and then click *OK* in the confirmation message window.

   The adapter is added to the *Configured* folder in the navigation pane.

Managing JD Edwards World Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the data types.

Note: To create synonyms, you need to refresh the metadata, and also have both the JD Edwards World adapter and a Db2 adapter configured.

Procedure: How to Refresh the Metadata

1. Right-click the configured JD Edwards World adapter and select *Refresh Metadata Repository*.

   The Refresh Metadata Repository pane opens.
2. In the JD Edwards World Version field, choose *7.x* or *8.x*.
3. Enter the *location* of each of the following JDE files in the space provided. Only the library name is needed.

For Version 7	For Version 8	File Use
F0004 Library	F0004 Library	Contains JDE user-defined code types.
F0005 Library	F0005 LIbrary	Contains JDE user defined codes file.
F9202 Library	F9202 Library	Contains JDE data field specifications.
F9201 Library	F9210 Library	Contains JDE data field display text.
UDC Library		Contains the name of the new library to be created. It is recommended that you start the name with UDC*xxx*.

4. Click *Refresh Now* to create the files.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter.

1. From the Web Console *Applications* page, click *Get Data*.
2. Right-click the configured *JD Edwards World* adapter and select *Create Synonym*.

   The Select connection to create synonym page opens.
3. Click the Db2 connection to be used.
4. Complete the information required using the *Synonym Creation Parameters for JD Edwards World* on page 1231
5. The final step is setting JD Edwards World security options.

   For details, see *How to Set JD Edwards World Security* on page 1235.

Reference: Synonym Creation Parameters for JD Edwards World

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select Synonym Candidates for Db2 (DB2CON1) Step 1 of 3

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Library

Type a string for filtering the Library (or Db2 Collection), inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner IDs begin with the letters ABC; %ABC to select tables or views whose owner IDs end with the letters ABC; %ABC% to select tables or views whose owner IDs contain the letters ABC at the beginning, middle, or end.

Object Name

Type a string for filtering the table, view, or object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all tables, views, or objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Note: When you create a synonym for Db2 on the IBM i platform, standard IBM i naming conventions apply to the target data source. Therefore, the Adapter for Db2 supports the use of double-quotation marks around any library name and/or file name that contains lowercase or NLS characters.

Create Synonym for Db2 (DB2CON1) Step 2 of 3

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

One-part name

On the IBM i platform, the *One-part name* check box is unchecked by default. The unchecked behavior generates a table name that includes the explicit name of the library containing the table. For example, if you specified a library on the first Create Synonym pane, a qualified name like the following is automatically created in the Access File:

```
TABLENAME=MYLIB/MYTABLE
```

With this explicit type of entry in the Access File, at run-time the library is directly located and searched for the table name. If you select the check box, the explicit library name is not stored in the metadata (Access File). When the synonym is generated, the library portion of the table name is omitted from the Access File, and appears as follows:

```
TABLENAME=MYTABLE
```

With this type of entry in the Access File, at run time the library path of the user is searched until the table name is located.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Create Synonym for Db2 Step 3 of 3

Select date format

The options are: YMD, YYMD, DMY, MDY, MDYY, DMYY, MYY,YYM. (YYMD is the default setting.) The selected format will be used only if the field is described as a DATE in the DataDictionary.

Presumptive Join

Select the *Presumptive Joins* check box to include additional DEFINEs (virtual fields) for presumptive join fields in the synonym.

The default setting is *ON*.

Field Names

Select the *Long Fieldname* radio button (the default) to display the field descriptions as names on reports. Select the *Short Fieldname* radio button to use the JDE aliases as field names on reports. The default setting is *Long Fieldname*.

Language Code

Enter the appropriate Language Code, which exists in the JDE F9292 file. (Leave the field blank for English.)

UDC

Select the *UDC* check box to ensure that UDC description fields are generated as DEFINEs (virtual fields) in the synonym. The default setting is *ON*.

Combine UDC

Select the *Combine UDC* check box to Combine User Defined Code. The default setting is *OFF*.

Enabling JD Edwards World Security

You can set security options after you configure the adapter from the Web Console.

Procedure: How to Set JD Edwards World Security

1. From the list of configured adapters in the navigation pane, right-click *JD Edwards World*, and select *Properties*.

   The Change Connect Parameters pane for JD Edwards World opens.

2. You can accept the defaults (unchecked) or select one or more options by clicking the corresponding check boxes.

   Business Unit Security

   Check this box to enable automatic execution of JD Edwards World Business Unit Security. The server for IBM i automatically restricts user access to data, based on information retrieved from the F0001 and F0006 tables, and then adds appropriate WHERE conditions to the users submitted data access request.

   Unchecked (OFF) is the default setting.

   If you check this parameter, you cannot turn it OFF until the server is shut down and then restarted (with no parameter settings).

   Search Type Security

   Check this box to enable automatic execution of JD Edwards World Search Type Security. The server for IBM i automatically restricts user access to data, based on information retrieved from the F0005 table, and then adds appropriate WHERE conditions to the submitted data access request of the user.

   Unchecked (OFF) is the default setting.

   If you check this parameter, you cannot turn it OFF until the server is shut down and then restarted (with no parameter settings).

   Business Unit (for PA) Security

   Check this box to revert (if necessary) to an older security model used by this adapter.

   Unchecked (OFF) is the default setting.

   If checked, this option overrides standard Business Unit Security (as described above).

   Column Security

   Check this box to enable column security based on information in the F9401 file.

   Unchecked (OFF) is the default setting.

If you check this parameter, you cannot turn it OFF until the server is shut down and then restarted (with no parameter settings).

Select Profile

In Web Query this must be EDASPROF.

3. Click the *Configure* button to complete the process.

Enabling Tracing

You can easily trace any problem that arises using the Web Console tracing facility. For information about Trace options, see *Tracing Server Activity* in the *Server Administration* manual.

Frequently Asked Questions

Which Application directory should I store my JDE Master and Access Files in?

For a full WebFOCUS installation, we highly recommend storing all Master and Access Files in a separate Application directory from where the server was installed. You can then add the directory to the server's path prior to starting the server. This will not only avoid confusion about where these files reside, but will also make maintenance easier during server upgrades.

I've been using JD Edwards World under a 5.x server and I'm upgrading to a newer server level. Will the Master Files I built under the 5.x server work with 7.x?

Existing Master Files built with PSWCONV from the 5.x Server are *not* supported by the 7.x Server.

Enhancements have been made to the adapter conversion JDE procedure to accommodate the 7.x Server architecture. We suggest saving your existing 5.x Master Files as a backup, and recreating the metadata with the new version of the Adapter for JD Edwards.

Chapter 48

Using the Adapter for Jethro

Jethro is an acceleration engine that makes real-time Business Intelligence work on Big Data.

The Adapter for Jethro is available in the following modes:

- ODBC (on Windows only)
- JDBC (on Linux/Unix/Windows)

Only one mode (JDBC or ODBC) can be configured for a server instance.

In this chapter:

Obtaining the Jethro Drivers

The JDBC and ODBC drivers for Jethro can be downloaded from *https://jethro.io/driver-downloads*.

If you are using the ODBC driver, download the version (32 or 64 bit) that corresponds to the DataMigrator or WebFOCUS Reporting Server that you are running.

Preparing the Environment for the JDBC Driver

The following components are needed to use the JDBC adapter for Jethro:

- **Java.** To use this Java-based adapter, you must have Java, Version 1.7 or later, installed. Java can be downloaded from *http://www.java.com*.

  The location of the Java must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/lib/jvm/jre-1.7.0
  ```

If you have JDK installed:

```
export JAVA_HOME=/usr/lib/jvm/jdk-1.7.0
```

If you are using Windows, right-click *Computer* and click *Properties*. Then click *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

```
C:\Program Files\Java\jdk7\bin\server;C:\Program Files\Java\jdk7\bin;
```

- **JDBC Driver**

  The JDBC driver consists two jar files (JethroDataJDBC.jar and protobuf-java-2.4.1.jar). They are included in the Jethro distribution at $JETHRO_HOME/jdbc and available for download from *https://jethro.io/driver-downloads*. If you are installing the server on the same system where Jethro is installed, you can point to the jar file as described in the next section. If you are installing the server on some other system, copy the file to a location of your choice.

Procedure: How to Configure the Java CLASSPATH

The location of the Jethro JDBC Driver must be specified to the server. If you are running the server on the same system as a Drillbit, you can specify its location. If you are running the server on another system, copy the files listed below to some location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console menu bar, click *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.

2. Expand the *Java Services* folder. Right-click *DEFAULT* and click *Properties*.

   The Java Services Configuration page opens.

3. Click the chevrons to expand Class Path.

   In the IBI_CLASSPATH box, enter the full location of the jar file. For example:

   ```
   /opt/jethro/current/JethroDataJDBC.jar
   /opt/jethro/current/protobuf-java-2.4.1.jar
   ```

   Note: The file names must be entered one per line.

If you are installing the adapter on a different system than where you have Apache Phoenix installed, copy the jar file to a location on that system.

Note: For a server running on Windows, use Windows syntax for directory names. For example:

```
C:\jethro\JethroDataJDBC.jar
C:\jethro\protobuf-java-2.4.1.jar
```

4. Scroll down and click the *Save and Restart Java Services* button.

Configuring the JDBC Adapter for Jethro

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Jethro JDBC Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the Jethro folder.
5. Click *Configure ODBC*.
6. In the URL box, enter the URL used to connect to your Drillbit. For more information, see *Jethro URL Configuration Settings* on page 1240.
7. In the Driver Name box, enter the following:

   ```
   com.jethrodata.JethroDriver
   ```

8. Enter your User and password. The default values are jethro/jethro.
9. Click *Test*. You should see a list of data sources on your server.

10. Click *Configure*.

Reference: Jethro URL Configuration Settings

The Adapter for Jethro is under the SQL group folder.

The URL configuration for the Jethro adapter is:

```
jdbc:JethroData://host:port[;host:port]…/instance_name
```

where:

host

Is the name or IP addresses where the Jethro server is running.

port

Is the port number for Jethro. The default value is 9111.

instance_name

Is the name of the Jethro server instance.

Troubleshooting

If the server is unable to configure the connection, a dialog box opens with an error message. The message typically begins with the following:

```
(FOC1400) SQLCODE IS -1 (HEX: FFFFFFFF) XOPEN: nnnn
```

where:

nnnn

Is the message number returned.

Some of the more common errors are:

```
(FOC1500) : (-1) [00000] JDBFOC>> connectx():
(FOC1500) : java.lang.UnsupportedClassVersionError:
org/apache/drill/jdbc/Driver :
(FOC1500) : Unsupported major.minor version 51.0
(FOC1479) ERROR CONNECTING TO SQL DATABASE
```

This indicates that your Java version is less than 1.7, which is required.

Configuring the ODBC Adapter for Jethro

Configuring the adapter consists of specifying connection and authentication information for the connection that you want to establish.

Note that the ODBC Adapter for Jethro is only available for servers running on Windows.

Procedure: How to Configure the ODBC Adapter for Jethro

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

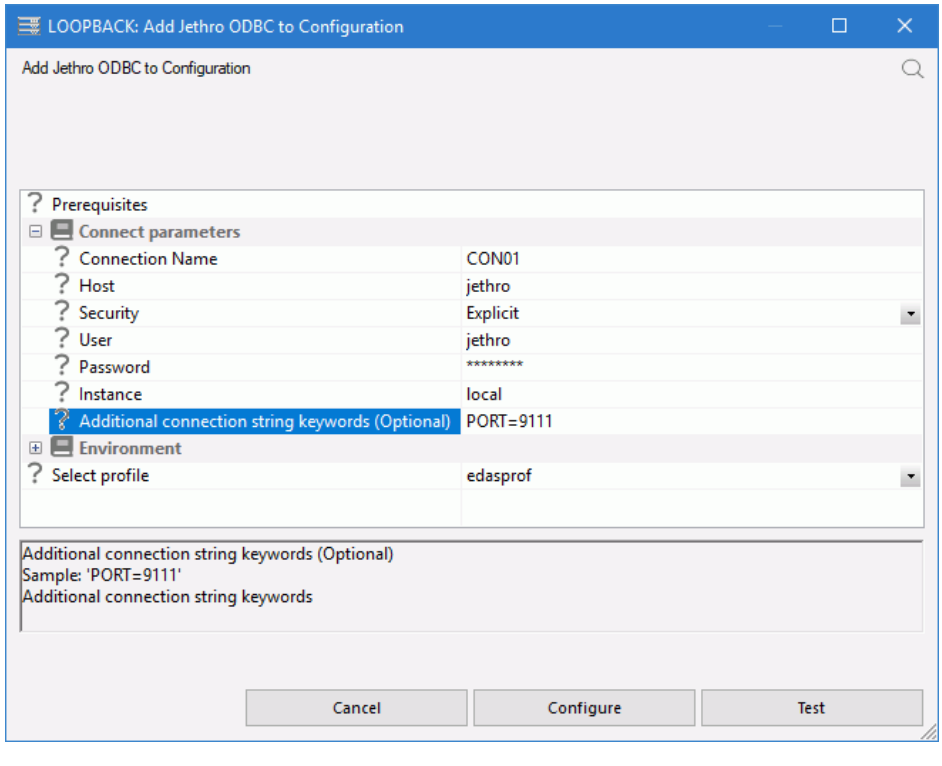
   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. Right-click *Jethro* and click *ODBC*.

   The Add Jethro ODBC to Configuration dialog box opens, as shown in the following image.



4. Enter the values for the parameters as described in *Connection Attributes for the ODBC Adapter for Jethro* on page 1242.
5. Click *Configure*.

Reference: Connection Attributes for the ODBC Adapter for Jethro

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name
Logical name used to identify this particular set of connection attributes. The default is CON01.

Host
Name of the Host where the Server is running.

Security
Select *Explicit* from the pull-down menu. Type the user ID (jethro by default) and password (also jethro by default) for your Jethro server.

Instance
Type the instance name for your Jethro server. For example, demo.

Additional Connection-String Keywords
Type PORT=*port_number*, where *port_number* is where your Jethro server is listening, 9111 by default.

Creating Synonyms With Jethro

Synonyms define unique names, or aliases, for each Jethro table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Jethro table.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

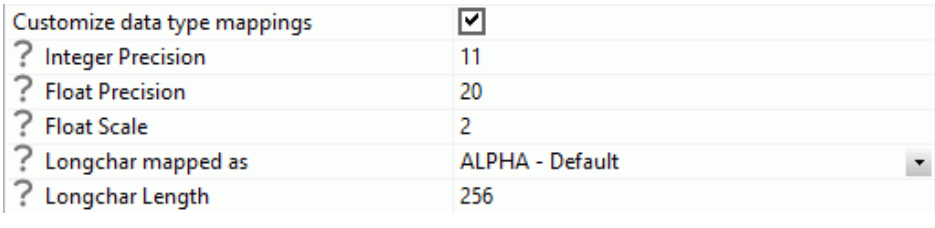
Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.
5. Optionally, expand the *Customize data type mappings* section and enter a value for CONV_LONGCHAR_LEN or numeric precision, as shown in the following image.



6. Select the check box for the objects that you want to create synonyms for. If you want to change the name of the synonym from the default name, click the name and edit it as needed.
7. Click *Create Synonym*.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Chapter 49

Using the Adapter for JSON

The Adapter for JSON (JavaScript Object Notation) allows applications to access JSON documents as data sources. The adapter traverses the hierarchical trees and returns an answer set to the requesting application.

In this chapter:

Preparing the JSON Environment

The Adapter for JSON does not require any environment variables to be configured.

Configuring the Adapter for JSON

You can configure the Adapter for JSON from the Web Console or the Data Management Console.

Procedure: How to Configure the Adapter

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Note: The JSON adapter is in *XML-based* drop-down.. No input parameters are required.

Associating a Master File With a JSON Document

You must explicitly issue a FILEDEF command to associate a Master File with a JSON document. In order to properly code the FILEDEF command, you need to determine how the JSON documents are organized. If the documents are all in one file, the organization is called a COLUMN. If there are various documents in one directory, then the organization is called a COLLECTION.

Source	Collection Organization	Column Organization
disk	supported	supported
http	not supported	supported
https	not supported	supported

Syntax: How to Associate a Master File With a JSON Document

```
FILEDEF ddname source filename
```

where:

ddname

Is the logical reference name matching the desired Master File.

source

Is the location of the JSON document. Possible values are http, https, and disk.

filename

For a JSON Column, is the full path to the source file.

For a JSON Collection, is the full path to the source directory. The directory name must be followed by the wildcard characters '*.*'.

The file name must start with 'http://' or 'https://' if the JSON documents are to be read using an HTTP or HTTPS session.

Example: Issuing a FILEDEF Command for a JSON Column

Note: If you are working on the Web Console or Data Management Console Create Synonym panes, see *Synonym Creation Parameters for JSON* on page 1249 for details about how FILEDEFs are created.

For Windows, when you have the ordercol.JSON document under C:\, use the following FILEDEF in your profile:

```
FILEDEF ordercol disk C:\ordercol.JSON
```

For UNIX, when you have the ordercol.JSON document under /usera/JSON/, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK /usera/JSON/ordercol.JSON
```

For z/OS, when you have the ordercol.JSON document in TEST.JSON, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.JSON'
```

For HTTP and HTTPS, when you have the ordercol.JSON document under http://www.test.com/JSONtest/ (or https://www.test.com/JSONtest/), use the following FILEDEF in your profile:

```
FILEDEF order http http://www.test.com/JSONtest/ordercol.JSON
```

or

```
FILEDEF order http https://www.test.com/JSONtest/ordercol.JSON
```

Example: Issuing a FILEDEF Command for a JSON Collection

Note: If you are working on the Web Console or Data Management Console Create Synonym panes, see *Synonym Creation Parameters for JSON* on page 1249 for details about how FILEDEFs are created.

For Windows, when you have the order1.JSON and order2.JSON documents under C:\ORDERS, use the following FILEDEF in your profile:

```
FILEDEF order disk C:\orders\*.*
```

For UNIX, when you have the ordercol.JSON document under /usera/orders, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.JSON'
```

For z/OS, when you have the order document in TEST.JSON, use the following FILEDEF in your profile:

```
FILEDEF order DISK /usera/orders/*.*
```

Managing JSON Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the JSON data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each JSON data structure that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given JSON document.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- ❑ **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- ❑ **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for JSON

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

You can create a synonym based on a JSON document:

- ❑ Select a document instance (Step 1) and click *Create Synonym based on Document Instance*.

Select Document Instance parameters (Step 1 of 2)

Use HTTP URL

Enables you to select a document instance from a URL. This selection requires a Base Location, Document Name, and Document Extension.

Base Location

Defines the location of the document instance.

- If *Use HTTP URL* is not selected, enter a physical path or application directory and the JSON document name, or click the ellipsis (...) to navigate to the document.
- If *Use HTTP URL* is selected, enter the http address of a directory that contains the JSON document you are using to create the synonym. (This functionality is not available when the JSON document is a local file.) The URL must start with http:// or https://.

Document Name

Enter the name of the JSON document.

Document Extension

Enter the document extension. The default is JSON.

Create Synonym based on Document Instance

Clicking this button enables you to create the synonym from the document instance.

Create Synonym for JSON (Step 2 of 2)

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Synonym Name

Indicates the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Accessing JSON Documents From a Relational DBMS JSON Data Type

JSON documents might be stored in any fields or columns in any data source. Reporting from such documents is supported by defining their structure as subtrees attached to a parent segment which describes the original data.

The synonym creation process must be run against the data in the DBMS and against the JSON document. The two Master Files must then be combined to make the JSON Master File a child of the Master File created against the DBMS. A FILEDEF is not needed in this instance.

Procedure: How to Access JSON Data From an RDBMS Using Web Console or Data Management Console Tools

1. Using the Web Console or the Data Management Console Create Synonym facility, generate a synonym for an RDBMS data source that contains a column of JSON data. Regardless of the data type used to contain the JSON data in the native data source, it will be mapped as a TX column in the Master File synonym. (For many RDBMS engines, the CLOB data type is mapped to TX.)
2. Open the generated Master File in the Synonym Editor. The Master File appears in the right pane in Text View. For example, the Master File for a Db2 data source might look as follows:

```
FILENAME=TESTJSON, SUFFIX=DB2     , $
SEGMENT=TESTJSON, SEGTYPE=S0, $
FIELDNAME=STORE, ALIAS=STORE, USAGE=A30, ACTUAL=A30, $
FIELDNAME=ADDRESS, ALIAS=ADDRESS, USAGE=A20, ACTUAL=A20,
MISSING=ON, $
FIELDNAME=GLOSS, ALIAS=GLOSS, USAGE=TX50, ACTUAL=TX,
MISSING=ON, $
```

3. From the DMC, right-click a column described as TX that contains JSON data. The pop-up menu contains the *Pivot* option.
4. Select *Pivot*, then *Multiple values to rows*. This option reads the JSON column directly and creates a new segment with SEGSUF=JSON.

   The *Select mapping source* dialog box opens.

   a. Select *Rows to sample for structural analysis* and click *Next*.

      The *Column mapping* dialog box opens displaying two rows from the column that is being pivoted.

      Examine the data to make sure it looks like JSON.

   b. In the *Select data type* pull-down list, select *JSON* and click *Apply*.

      The resulting Master File contains the definition of the JSON data, represented as a new DUMROOT segment with SEGSUF=JSON, which appears in the Text View pane following the original RDBMS segment.

      You can expand DUMROOT to see the names of the fields found in the JSON.

      You can right-click DUMROOT to see the values of those fields.

The Master File might now look as follows, with the DUMROOT segment containing the data from the JSON column:

```
FILENAME=testjson, SUFFIX=DB2     , $
  SEGMENT=TESTJSON, SEGTYPE=S0, $
    FIELDNAME=STORE, ALIAS=STORE, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=ADDRESS, ALIAS=ADDRESS, USAGE=A20, ACTUAL=A20,
      MISSING=ON, $
    FIELDNAME=GLOSS, ALIAS=GLOSS, USAGE=TX50, ACTUAL=TX,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=DUMROOT, SEGTYPE=S0, SEGSUF=JSON    ,
      PARENT=TESTJSON, POSITION=GLOSS, $
    FIELDNAME=DUMROOT, ALIAS=JSON_DUMMY_EL, USAGE=A1, ACTUAL=A1,
        ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=TITLE, ALIAS=title, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSARY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSDIV, ALIAS=GlossDiv, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSARY, PROPERTY=ELEMENT,  $
    FIELDNAME=TITLE1, ALIAS=title, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDIV, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSLIST, ALIAS=GlossList, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSDIV, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSENTRY, ALIAS=GlossEntry, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSLIST, PROPERTY=ELEMENT,  $
    FIELDNAME=ID, ALIAS=ID, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=SORTAS, ALIAS=SortAs, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSTERM, ALIAS=GlossTerm, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
```

```
    FIELDNAME=ACRONYM, ALIAS=Acronym, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=ABBREV, ALIAS=Abbrev, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSDEF, ALIAS=GlossDef, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=PARA, ALIAS=para, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDEF, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSSEE, ALIAS=GlossSee, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
  SEGMENT=GLOSSSEEALSO, SEGTYPE=S0, PARENT=GLOSSARY, $
    FIELDNAME=GLOSSSEEALSO, ALIAS=GlossSeeAlso, USAGE=A55,
ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDEF, PROPERTY=ELEMENT,  $
```

5. From the Synonym Editor's File menu, save the updated Master File.

Procedure: How to Access JSON Data From an RDBMS Manually

Suppose that you have a table in an RDBMS with one or more columns storing JSON data. In order to report from the JSON data, following these steps:

1. Create the Master File for the relational data source using the format for that DBMS.

```
FILENAME=TESTJSON, SUFFIX=DB2     , $
  SEGMENT=TESTJSON, SEGTYPE=S0, $
    FIELDNAME=STORE, ALIAS=STORE, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=ADDRESS, ALIAS=ADDRESS, USAGE=A20, ACTUAL=A20,
      MISSING=ON, $
    FIELDNAME=GLOSS, ALIAS=GLOSS, USAGE=TX50, ACTUAL=TX,
      MISSING=ON, $
```

2. Create a Master File for the JSON document in the column of the RDBMS table. If there are two JSON documents with different formats, you must create a Master File for each one.

3. Manually combine the Master Files. On each root segment for the JSON Master File, add three fields: position, parent, and segsuf. The POSITION keyword identifies the field containing the JSON document. The PARENT field describes the original data source. The field SEGSUF defines the root segment of a JSON document representing sub-tree. The total length of all fields in the Master File must not exceed the FOCUS limitation of 32k. If it does, the query will fail.

```
FILENAME=BASEAPP_GLOSSARY, SUFFIX=JSON    ,
 DATASET=c:\json\glossary.json, $
  SEGMENT=GLOSSARY, SEGTYPE=S0, $
    FIELDNAME=GLOSSARY, ALIAS=glossary, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=TITLE, ALIAS=title, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSARY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSDIV, ALIAS=GlossDiv, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSARY, PROPERTY=ELEMENT,  $
    FIELDNAME=TITLE1, ALIAS=title, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDIV, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSLIST, ALIAS=GlossList, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSDIV, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSENTRY, ALIAS=GlossEntry, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSLIST, PROPERTY=ELEMENT,  $
    FIELDNAME=ID, ALIAS=ID, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=SORTAS, ALIAS=SortAs, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSTERM, ALIAS=GlossTerm, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=ACRONYM, ALIAS=Acronym, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $

    FIELDNAME=ABBREV, ALIAS=Abbrev, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSDEF, ALIAS=GlossDef, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=PARA, ALIAS=para, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDEF, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSSEE, ALIAS=GlossSee, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
  SEGMENT=GLOSSSEEALSO, SEGTYPE=S0, PARENT=GLOSSARY, $
    FIELDNAME=GLOSSSEEALSO, ALIAS=GlossSeeAlso, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDEF, PROPERTY=ELEMENT,  $
```

Combined Master file:

```
FILENAME=testjson, SUFFIX=DB2     , $
  SEGMENT=TESTJSON, SEGTYPE=S0, $
    FIELDNAME=STORE, ALIAS=STORE, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=ADDRESS, ALIAS=ADDRESS, USAGE=A20, ACTUAL=A20,
      MISSING=ON, $
    FIELDNAME=GLOSS, ALIAS=GLOSS, USAGE=TX50, ACTUAL=TX,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=GLOSSARY, SEGTYPE=S0, SEGSUF=JSON    , PARENT=TESTJSON,
POSITION=GLOSS, $
    FIELDNAME=GLOSSARY, ALIAS=glossary, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=TITLE, ALIAS=title, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSARY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSDIV, ALIAS=GlossDiv, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSARY, PROPERTY=ELEMENT,  $
    FIELDNAME=TITLE1, ALIAS=title, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSDIV, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSLIST, ALIAS=GlossList, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSDIV, PROPERTY=ELEMENT,  $


FIELDNAME=GLOSSENTRY, ALIAS=GlossEntry, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSLIST, PROPERTY=ELEMENT,  $
    FIELDNAME=ID, ALIAS=ID, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=SORTAS, ALIAS=SortAs, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSTERM, ALIAS=GlossTerm, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=ACRONYM, ALIAS=Acronym, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=ABBREV, ALIAS=Abbrev, USAGE=A55, ACTUAL=A55,
      MISSING=ON,
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
    FIELDNAME=GLOSSDEF, ALIAS=GlossDef, USAGE=A1, ACTUAL=A1,
      MISSING=ON, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
```

```
  FIELDNAME=PARA, ALIAS=para, USAGE=A55, ACTUAL=A55,
    MISSING=ON,
    REFERENCE=GLOSSDEF, PROPERTY=ELEMENT,  $
  FIELDNAME=GLOSSSEE, ALIAS=GlossSee, USAGE=A55, ACTUAL=A55,
    MISSING=ON,
    REFERENCE=GLOSSENTRY, PROPERTY=ELEMENT,  $
SEGMENT=GLOSSSEEALSO, SEGTYPE=S0, PARENT=GLOSSARY, $
  FIELDNAME=GLOSSSEEALSO, ALIAS=GlossSeeAlso, USAGE=A55, ACTUAL=A55,
    MISSING=ON,
    REFERENCE=GLOSSDEF, PROPERTY=ELEMENT,  $
```

Conversion

The data in a JSON document may reflect dates or numeric values, however, all the fields in a Master File synonym are set to the ALPHA data type.

Numeric Values

In order to enable arithmetic operations on numeric fields, the data type specified in the USAGE attribute of a numeric field needs to be modified, depending on the data in the JSON document, to one of the following data types: Integer (I), Double Float (D), or Decimal (P). If the data type is modified to Double Float or Decimal, use scale and precision as necessary to describe the data in the JSON document.

Furthermore, it is recommended that the length of the ALPHA data type specified in the ACTUAL attribute of the numeric field be modified to reflect the maximum length of the data in the JSON document.

Dates in JSON

In order to enable arithmetic operations on dates, the data type specified in the USAGE attribute of a date field needs to be modified, depending on the date format used in the JSON document, to one of the following data types: YYMD, MDYY, or DMYY.

Furthermore, the length of the ALPHA data type specified in the ACTUAL attribute of the date field needs to be modified to 10.

Note:

- Once you have set the data type in the Master File to one of the date data types, you must make sure that in each of the JSON documents from which you report the date format is the same as the one you defined in the Master File. If the data types differ, an error will result.
- The date format in the answer set is not derived from the date format used in the JSON document and is always set to YYYY/MM/DD.

Example: Using Dates in JSON

If in the JSON document you have the following format:	Then use USAGE=
1996-01-30	YYMD
01-30-1996	MDYY
30-01-1996	DMYY

Chapter 50

Using the Adapter for Kafka

The Adapter for Kafka provides access to and reporting against messages resident in the Apache Kafka environment.

In this chapter:

Preparing the Kafka Environment

The Adapter for Kafka can be configured with or without support for Apache Avro, which is a data serialization system. It uses a schema to perform serialization and deserialization of Kafka messages.

The Adapter for Kafka without support for Apache Avro requires the kafka-clients jar file and the slf4j-api Jar file as part of the configuration process. Once you download the Kafka binaries from the Apache Kafka download site (*https://kafka.apache.org/downloads*), the necessary .jar files are in the *lib* directory.

You must add the .jar files to the IBI_CLASSPATH variable when you configure the adapter.

The Adapter for Kafka with support for Apache Avro requires a different set of .jar files to be configured in IBI_CLASSPATH, in order to deserialize the messages being polled from the Kafka environment. These .jar files are included as part of the Confluent Platform download (*https://www.confluent.io/download/*).

You can also configure the Adapter for Kafka in a Docker container.

Complete prerequisites and examples of configuration options are included in the *Prerequisites* link on the right-click menu for the adapter.

Configuring the Adapter for Kafka

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Adapter for Kafka

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click *Kafka* and click *Configure*.

Reference: Connection Attributes for Kafka

The Adapter for Kafka is under the Files group of adapters.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Basic Connection Attributes

Describes the connection attributes required for all connections to Kafka.

Connection Name

Is the logical name used to identify this set of connection attributes. The default is CON01.

Server

Is the host name of the Kafka host.

Port

Is the port on which Kafka listens, for example, 9092.

Security

There are three methods by which a user can be authenticated when connecting to the database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to Kafka, at connection time, for authentication as a standard login.
- **Password Passthru.** The user ID and password received from the client application are passed to Kafka, at connection time, for authentication as a standard login.
- **SASL GSSAPI.** The adapter connects to Kafka using Simple Authentication together with Security Layer and Generic Security Service Application Program Interface.

User

Is the primary authorization ID by which you are known to the data source. This field appears when the security selected is Explicit.

Password

Is the password associated with the primary authorization ID. This field appears when the security selected is Explicit.

JAAS Login File

Is the Java Authentication and Authorization Service Login Configuration File defining the user authentication and authorization services used by Kafka. This field appears when the security selected is SASL GSSAPI.

IBI_CLASSPATH

Lists search paths to required .zip and .jar files, each on a separate line. From the right-click menu for the adapter on the New Datasource list or the Configured Adapters list, select *Prerequisites* for more information.

Select profile

Select the profile in which to store the connection attributes. The default is edasprof, the server global profile.

Advanced HTTP Connection Options

Describes advanced security connection attributes.

SSL Configuration File

Is the Secure Socket Layer configuration file that defines the SSL configuration parameters used by Kafka.

TRUSTSTORE Location

Is the location of the file used to store certificates from trusted authorities.

TRUSTSTORE Password

Is the password required to read the contents of the TRUSTSTORE.

KEYSTORE Location

Is the location of the file used to store Private Key certificates.

KEYSTORE Password

Is the password required to read the contents of the KEYSTORE.

KEY Password

Is the password required to read the Private Key.

Managing Kafka Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Kafka data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each data structure that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Kafka topic.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

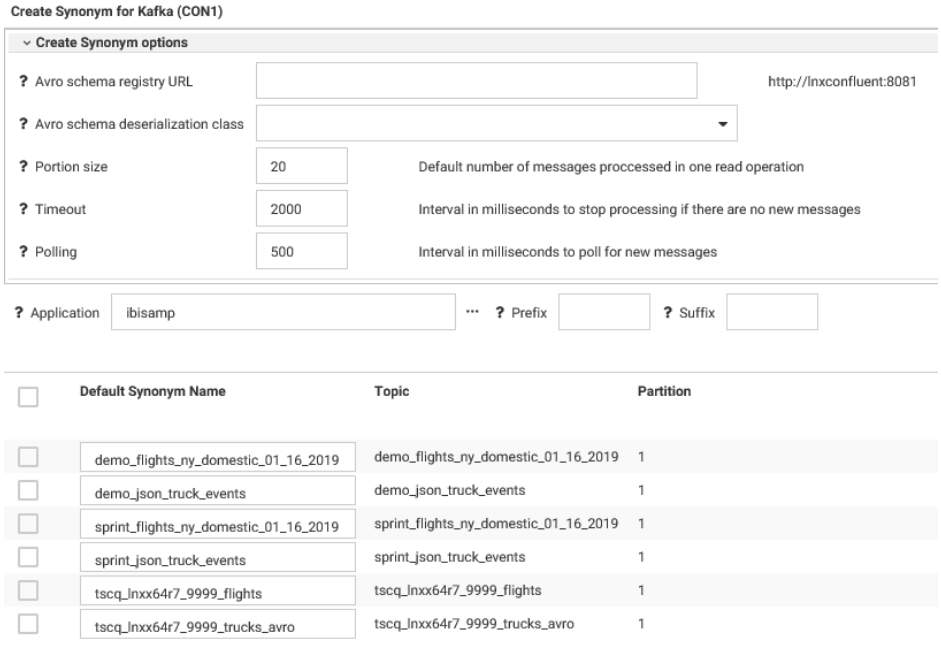
   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Kafka

The Create Synonym for Kafka page is shown in the following image.



The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. When you have configured the parameters and made the selections you need, click the *Create Synonym* button on the ribbon, which generates the synonym based on your entries.

To create synonyms for one or more Kafka topics, right-click a connection and click *Show Topics*. The Create Synonym for Kafka page opens on which you can select synonym objects and configure the following properties for topics within the Kafka environment.

Create Synonym for Kafka (Step 1 of 2)

Avro Schema Registry URL

Is the URL to the Apache Avro data serialization system.

Avro Schema Deserialization Class

Select the class used to deserialize the Avro Schema.

Portion Size

Is the default number of messages processed in one read operation. The default value is 20.

Timeout

Is the Interval, in milliseconds, to stop processing if there are no new messages. The default value is 2000.

Polling

Is the Interval, in milliseconds, to poll for new messages. The default value is 500.

Create Synonym for Kafka (Step 2 of 2)

Select the check boxes next to the topics for which you want to create synonyms.

Default Synonym Name

Indicates the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

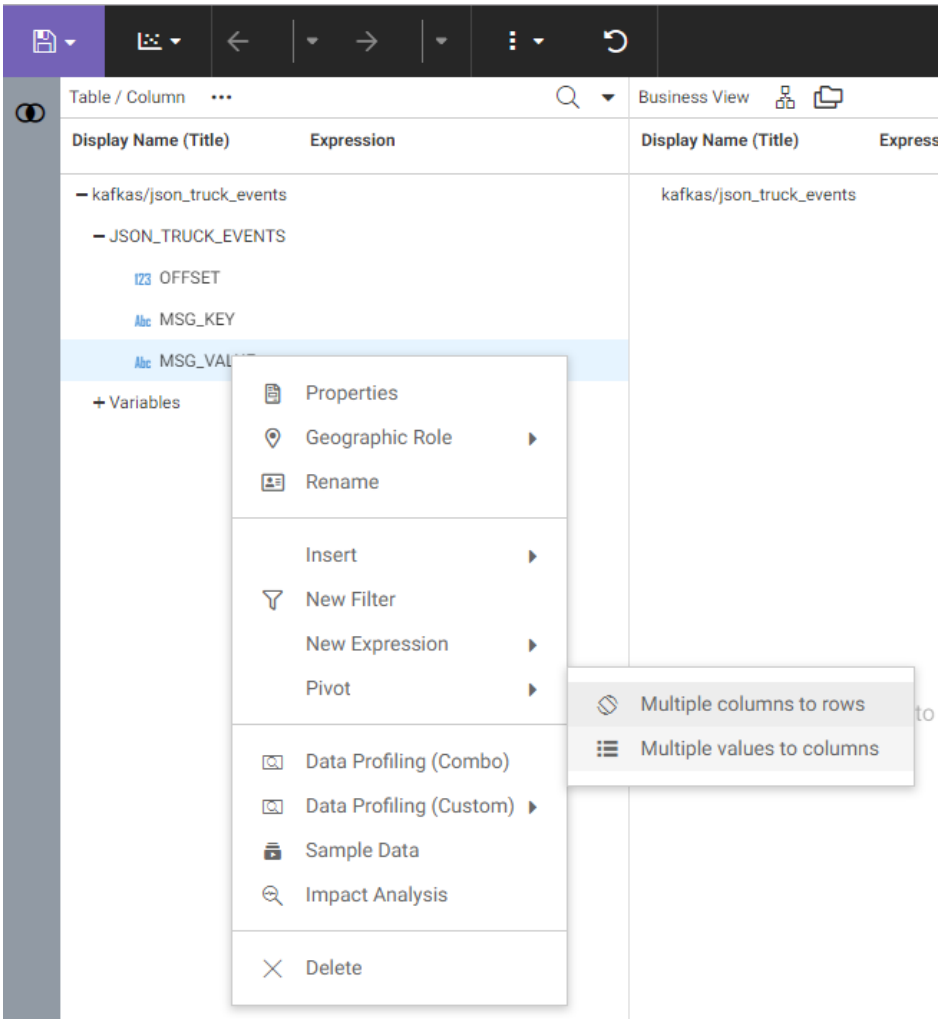
Reference: Managing Synonyms

Once you have created a synonym, you need to do the following to create a proper response segment dependent on the delimiter.

1. Open the synonym in the Synonym Editor by double-clicking the synonym name or right-clicking the synonym name and clicking *Open*.

   Depending on the Topic, the Message Value can delimit the components of each message in a variety of ways, for example, JSON, Pipe, or Comma. Pivoting this field will create a new segment in the synonym that has a field for each item within the Message Value.

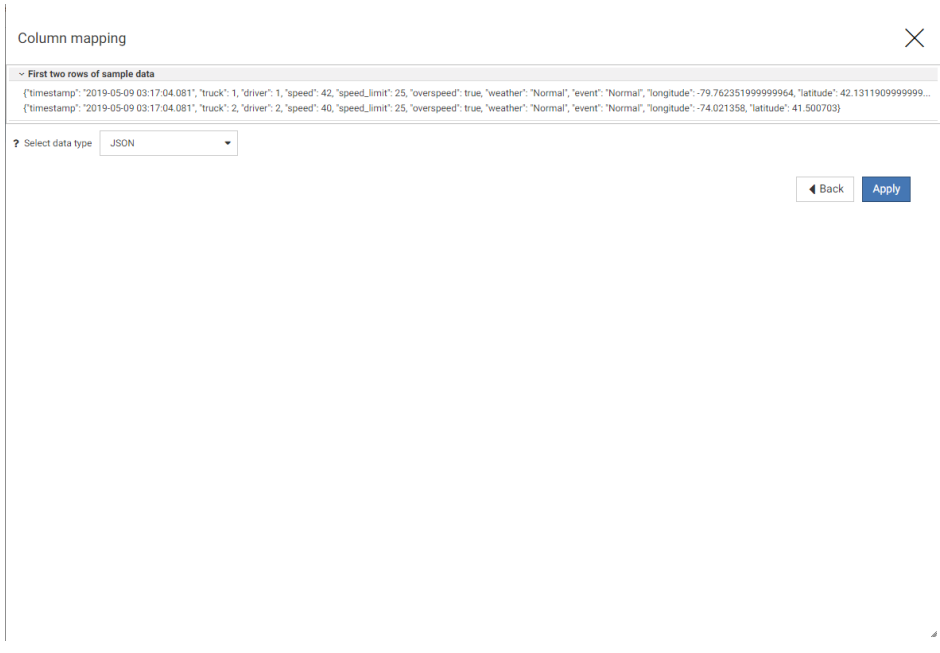
2. Pivot the field containing the message values by right-clicking the field, clicking *Pivot*, and clicking *Multiple values to columns*, as shown in the following image.



The Select Mapping Source page opens in which you can enter a number of rows to sample for structural analysis of the column to be pivoted. The default value is 1000.

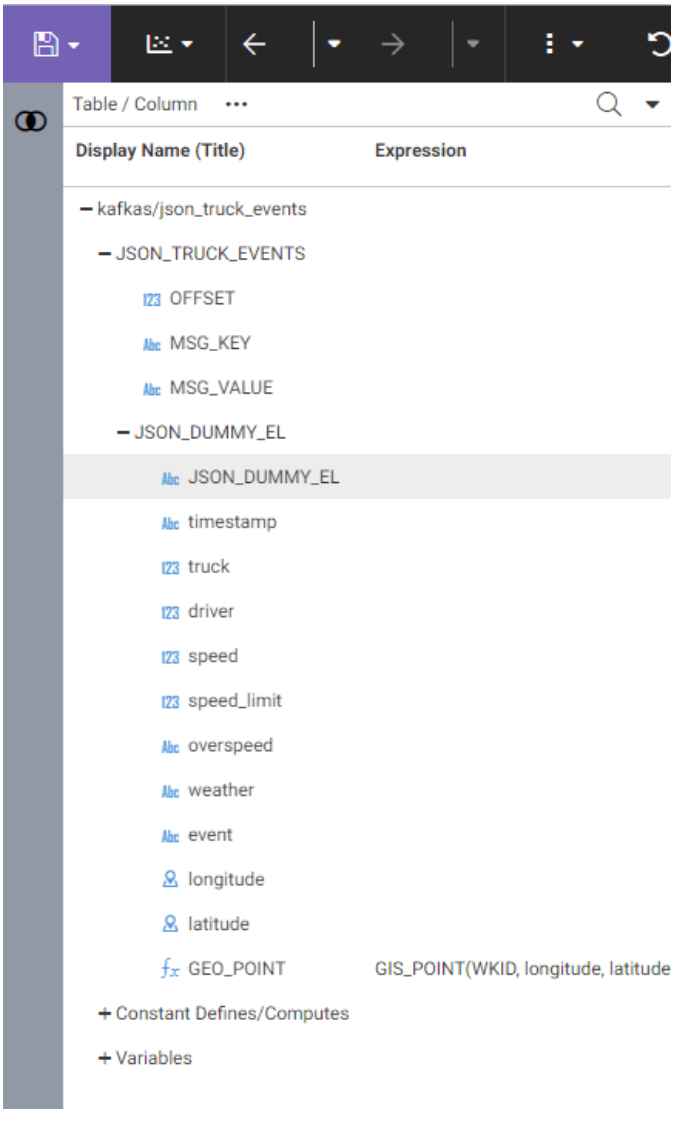
3. Click *Next*.

The Column Mapping page opens on which you can select the data type. A couple of rows of sample data are listed, as shown in the following image.



4. Click *Apply*.

A new segment is added to the synonym that lists each item from the JSON object as a separate field, as shown in the following image.



5. Save this synonym for use in reporting and messaging.

Chapter 51

Using the Adapter for Lawson

The Adapter for Lawson allows applications to access Lawson data sources.

The Adapter for Lawson automatically applies the appropriate Lawson security rules as defined in the Lawson LAUA repository. The adapter transparently generates dynamic DBA statements and/or dynamic WHERE clauses.

In this chapter:

Adapter for Lawson: Overview

The Adapter for Lawson works behind the scenes to control end users' access to the data in the Lawson system. When a user makes a request against a Lawson table, the Adapter for Lawson asks the following questions:

- Is this a Lawson File?
- Is this a Lawson User?
- Does the user have access to this table?

Once these questions have been answered, the Adapter for Lawson does the following:

- Dynamically generates DBA for any Application Security System Code, Record Level security, File Security, and Field Security. For more information on DBA, see the *Describing Data* manual.
- Generates WHERE clauses for any Lawson Company and Process Level, Record Security, and Field Security restrictions. These WHERE clauses are appended to any requests for data made to the Lawson data.

In order for the Adapter for Lawson to work properly, you must complete the following steps:

1. Verify that the server is installed.
2. Configure the Adapter for Lawson.
3. On the machine where Lawson is running, extract the Lawson security information into flat files using the lawsdmp.sh utility.
4. Copy or FTP the Lawson security information flat files to the server.
5. Refresh the security extracts to change the flat files into a format the server can understand.
6. Configure the data adapter and create the synonyms for the tables you plan to report from.

Configuring the Adapter for Lawson

You can configure the adapter using the Web Console or the Data Management Console (DMC).

Procedure: How to Declare Connection Attributes

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Lawson

The *Lawson* adapter is under the *ERP* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

DBMS Suffix

Specify the suffix of the database that Lawson is using. Possible values are:

`SQLORA` - Oracle

`SQLMSS` - Microsoft SQL Server

`SQLINF` - Informix

`SQLDB2` - Db2

Default Product Line

This value is customized by the site when the Lawson system is installed.

Server Authentication

- Check this box if the reporting server is secured. This option applies if every Lawson user has a user ID on the reporting server system.
- Leave the box unchecked if the reporting server is unsecured. This option applies if every Lawson user ID and password is stored and is being authenticated by Managed Reporting and/or a self-service application:
  - For Managed Reporting, the user ID must be the Lawson User ID.
  - For Self Service requests, the Lawson ID must be passed to the server.

  In either case, the IBI_REPORT_USER variable must be set to the Lawson User ID.

Windows-based Lawson?

Check this box if the Lawson application system is Windows-based.

Leave the box unchecked if it is UNIX-based.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Preparing the Lawson Environment

Preparing the Lawson environment involves extracting security information from Lawson, copying that information from flat files to the server, and refreshing the security extracts.

Important: Automatic Passthru must be disabled against Lawson data. To set Automatic Passthru off, enter the following command in the server profile, edasprof.prf:

```
SET APT = OFF
```

Procedure: How to Extract the Lawson Security Information

Perform the following steps on the machine where Lawson is running:

1. On the Lawson System, create the ibi_officer user ID, or any user ID with Security Officer capabilities. (This ID must be allowed to run the Lawson RNGDBDUMP command.)
2. Copy *lawsdmp.sh* and *lawsdmpu.sh* from the bin directory and *lawsfil.dat* from the etc directory of EDAHOME into the home directory of the ibi_officer.

   EDAHOME is the location of the files that run your server. The following table shows the default locations for EDAHOME.

Operating system	EDAHOME default directory location and name
Windows	\ibi\srv77\home
UNIX	/home/iadmin/ibi/srv77/home
IBM i (formerly known as i5/OS)	/home/iadmin/ibi/srv77/home

3. Change the execute attribute of lawsdmp.sh and of lawsdmpu.sh to *On*.
4. Schedule or run either lawsdmp.sh or lawsdmpu.sh to extract the latest changes to the Lawson Security System:

   - Use lawsdmp.sh for initial setup and when you wish to refresh all files. (This script extracts all necessary security information.)
   - Use lawsdmpu.sh when you wish to refresh *only* the user information file.

Procedure: How to Copy the Lawson Security Information Flat Files to the Server

1. Copy the lawsftp.dat file from the etc directory in EDAHOME to the lawsec app directory.
2. Edit lawsftp.dat to indicate the host name, user ID, and password of the remote machine that contains the extracts derived from running the lawsdmp.sh script.
3. If the server with the Adapter for Lawson is on the same machine as the Lawson System, copy the extracted files, *.ftm and *.cvs files, from the lawsdmp run home directory of ibi_officer to the Lawson Security Directory (the lawsec directory in APPROOT) of the server.

   If the server with the Adapter for Lawson is not on the same machine, run ftp with lawsftp.dat as input to ftp the extracted files to the Lawson Security Directory (the lawsec directory in APPROOT), as follows:

   ```
   ftp -niv < lawsftp.dat
   ```

Procedure: How to Refresh Security Extracts

1. If you have not already done so, start the server and the Web Console and/or Data Management Console (DMC).
2. Right-click the configured adapter on the Adapters navigation pane and choose *Refresh Security Extracts* from the menu.

This process creates the LAW* FOCUS files needed by the Adapter for Lawson to apply the necessary filtering to queries against the Lawson database files.

Managing Lawson Metadata

Once the Adapter for Lawson is configured and the environment properly set up, you must configure a data adapter (Db2, Microsoft SQL Server, Informix, or Oracle) to access the Lawson data and create synonyms for the associated database management system.

Updating Lawson Security Information

Every time you change Lawson security information, you need to update the server files.

The update process is similar to the one you use for the initial extraction of Lawson security information.

Procedure: How to Update Lawson Security Information

1. Run lawsdmp.sh (or lawsdmpu.sh) to extract the latest changes to the Lawson Security System in the ibi_officer user ID home directory.

2. If the server with the Adapter for Lawson is on the same machine as the Lawson System, copy the extracted files, *.ftm and *.cvs, from the lawsdmp run home directory of ibi_officer to the Lawson Security Directory (the lawsec directory in APPROOT) on the server. You can find these files in the lawsec directory.

   If the server is not on the same machine, run ftp with lawsftp.dat as input used to ftp the extracted files to the Lawson Security Directory (the lawsec directory in APPROOT). This should be done as follows:

   ```
   ftp -niv < lawsftp.dat
   ```

3. Start the server and the Web Console or Data Management Console.
4. From the Web Console menu bar, click *Adapters*

   or

   from the Data Management Console, expand the *Adapters* folder.

   The Adapters folder opens.
5. On the navigation pane, right-click the configured Lawson adapter and select *Refresh Security Extract* from the menu.

Chapter 52

Using the Adapter for LinkedIn

This section describes how to configure the LinkedIn Adapter.

In this chapter:

LinkedIn Adapter Overview

The LinkedIn Adapter is used to report against information resident in the LinkedIn environment. You can configure the LinkedIn Adapter using the Reporting Server Web Console. The adapter requires a connection, which stores the access token. A valid LinkedIn access token is needed to issue LinkedIn API calls. The token is associated with a LinkedIn application and a specific LinkedIn member. The access token is valid for 60 days after which a new access token would need to be obtained and configured.

The connection page shows permissions that will be granted to the LinkedIn application. You can uncheck permissions that should not be granted. The connection process will use only checked permissions. Authentication will take place even if no permissions are granted.

Creating a LinkedIn Application

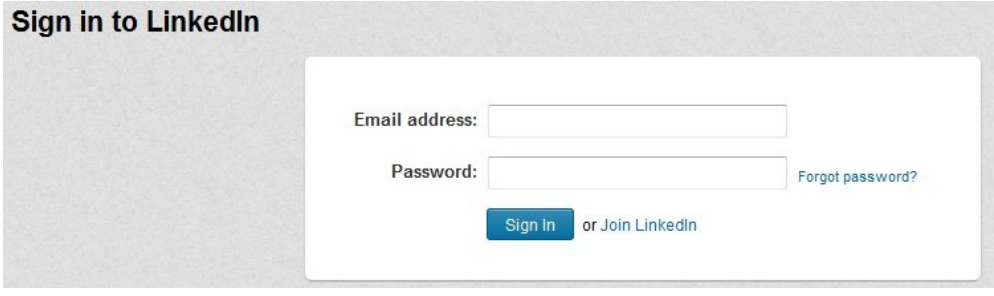
A LinkedIn application needs to exist before configuring the LinkedIn adapter

Procedure: How to Create a LinkedIn Application

1. From a browser, enter the following URL in a web browser:

   `https://www.linkedin.com/secure/developer`

A Sign in to LinkedIn screen opens, as shown in the following image.

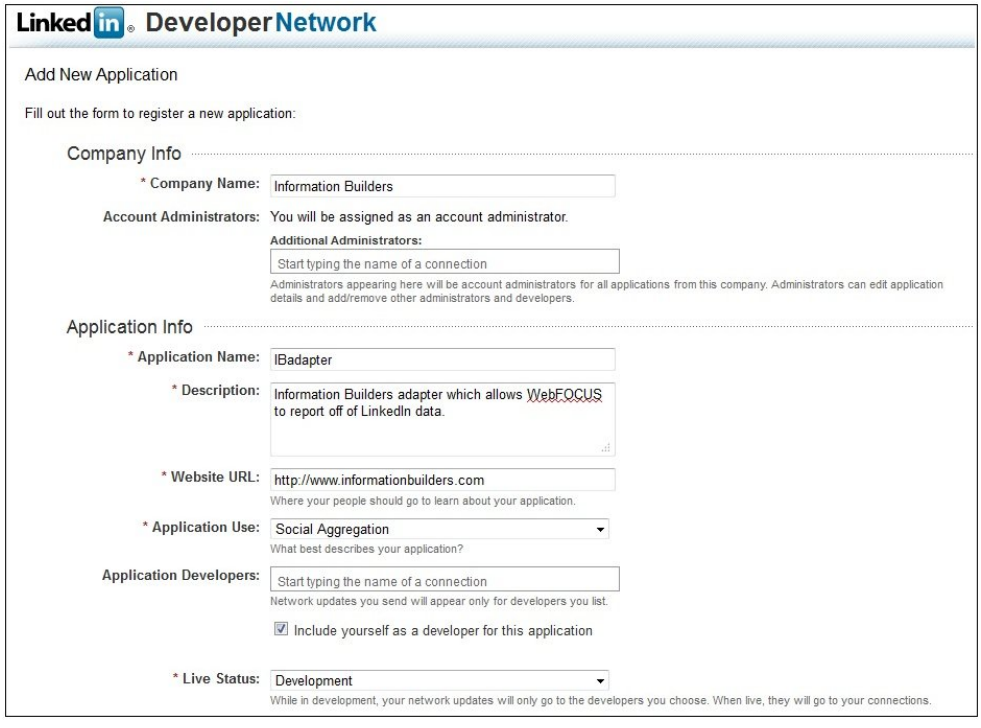


2. Enter the LinkedIn Sign in credentials and click *Sign In*.

   The LinkedIn Developer Network screen opens, as shown in the following image.

3. Click the *Add New Application* link.

The Add New Application screen opens, as shown in the following image.



4. Perform the following steps:

   a. Enter a company name hosting the new LinkedIn application in the Company Name field.

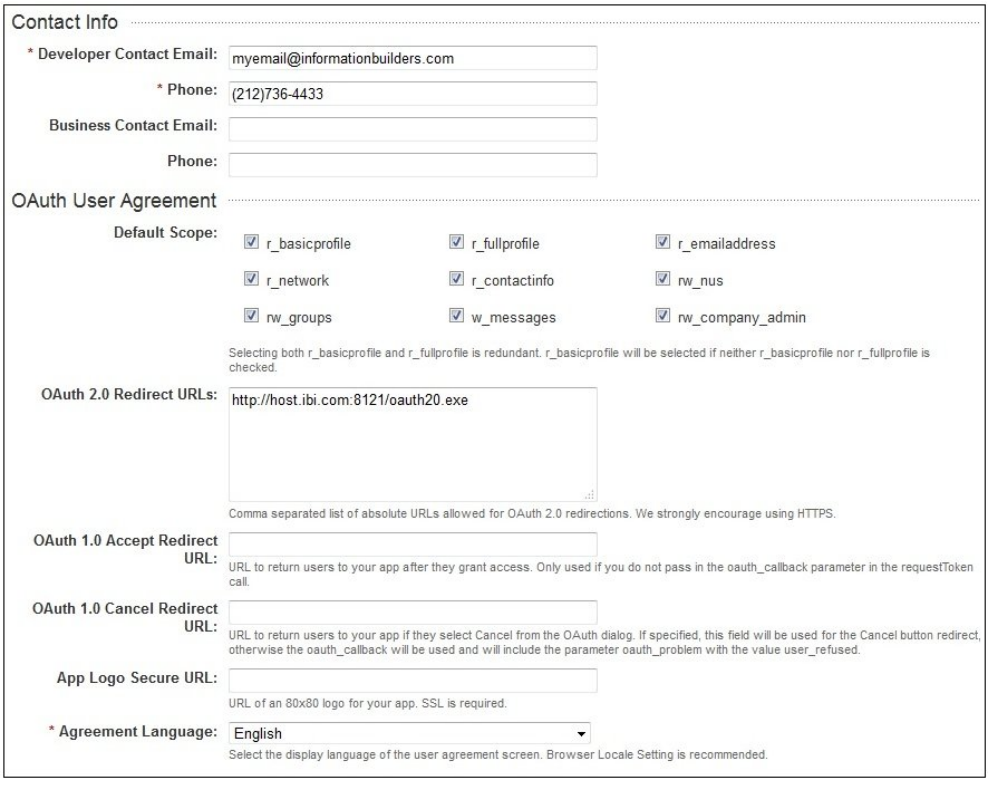
   b. Enter a name without spaces for the new LinkedIn application in the Application Name field.

   c. Enter a description for the new LinkedIn application in the Description field.

   d. Enter a web site URL for the company hosting the new LinkedIn application in the Website URL field.

   e. From the Application Use drop-down list, select a category which describes the use of the LinkedIn application.

5. Scroll down to the Contact Info section of the page, as shown in the following image.



6. Perform the following steps:

   a. Enter an email address for the LinkedIn application administrator in the Developer Contact Email field.

   b. Enter the telephone number for the LinkedIn application administrator in the Phone field.

   c. In the Default Scope section, check the permissions granted to the LinkedIn application.

   d. Enter the host name and port used to access the WebFOCUS Reporting Server Web Console with oauth20.exe in the OAuth 2.0 Redirect URLs field.

      For example:

      ```
      http://host.ibi.com:8121/oauth20.exe
      ```

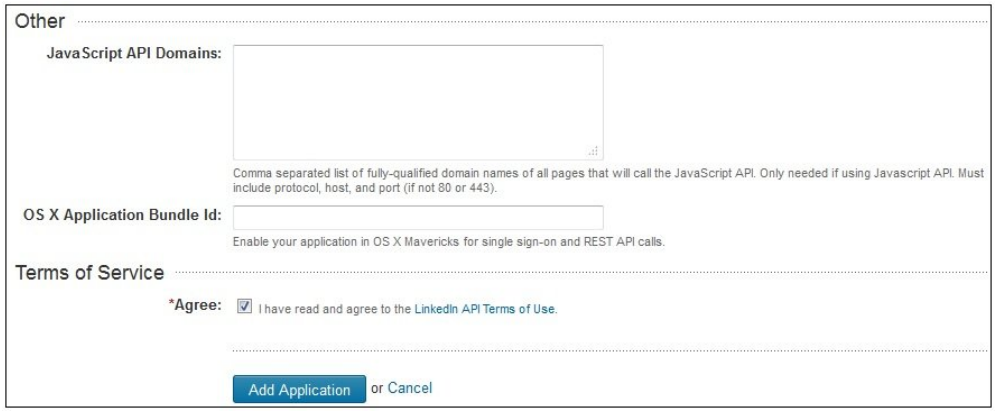
      If there are multiple WebFOCUS Reporting Servers used to access the LinkedIn application, separate each one of the OAuth 2.0 Redirect URLs with using a comma character (,).

For example:

```
http://host.ibi.com:8121/oauth20.exe,http://host2.ibi.com:8121/
oauth20.exe, http://localhost:8121/oauth20.exe
```

e. From the Agreement Language drop-down list, select the language to be used for the User Agreement screen.

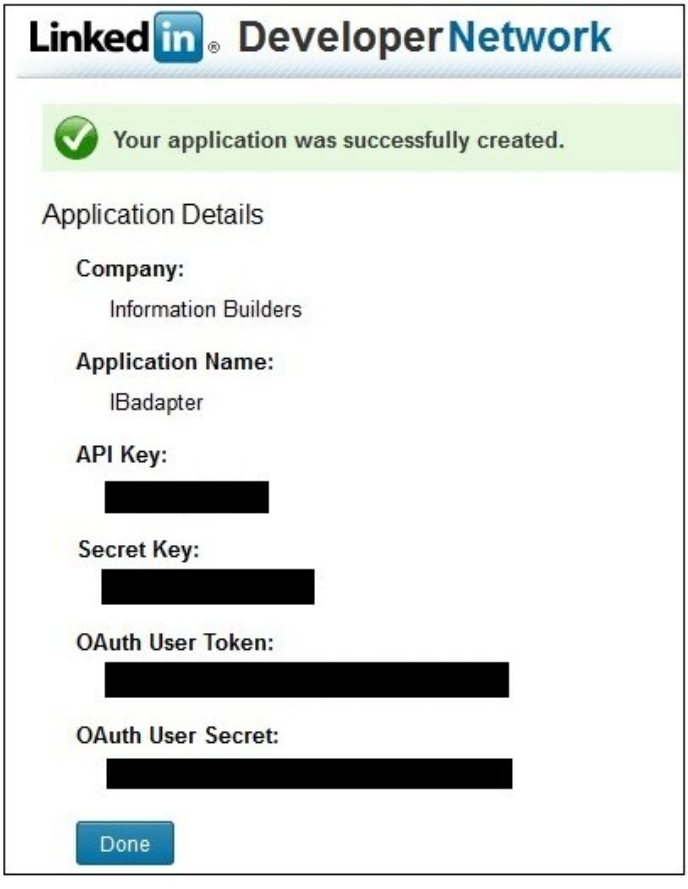
7. Scroll down to the Other section of the page, as shown in the following image.



8. Click the *LinkedIn API Terms of Use* link.

   If you accept the agreement, select *Agree* and then click *Add Application*.

The Application Details screen opens, as shown in the following image.



Note: The API Key and the Secret Key values will be required during the configuration of the LinkedIn adapter.

9. Click *Done*.

Configuring the LinkedIn Adapter

This section describes how to configure the LinkedIn Adapter.

Procedure: How to Configure the LinkedIn Adapter

1. Clear the cookies from the browser that will used to start the Reporting Server Web Console.

2. Access the WebFOCUS Reporting Server Web Console using the host name and port that you specified in the OAuth 2.0 Redirect URLs field of the LinkedIn application.

   For example:

   ```
   http://host.ibi.com:8121
   ```

   For more information on specifying values for the OAuth 2.0 Redirect URLs field, see *How to Create a LinkedIn Application* on page 1275.

3. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

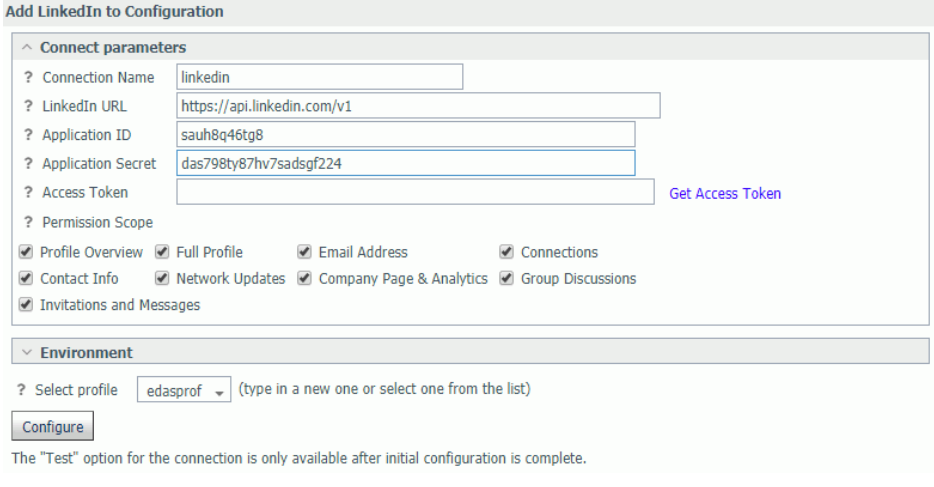
   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

4. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

5. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

6. Right-click the *LinkedIn* node and select *Configure*.

   The Add LinkedIn to Configuration pane opens, as shown in the following image.

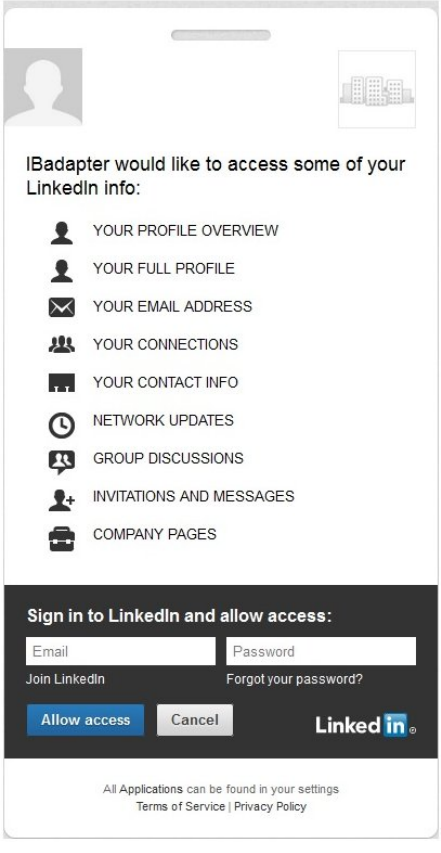


7. Enter the values for the Application ID and Application Secret as defined by the API Key and Secret Key respectively in the LinkedIn application.

   For more information, see *Creating a LinkedIn Application* on page 1275.

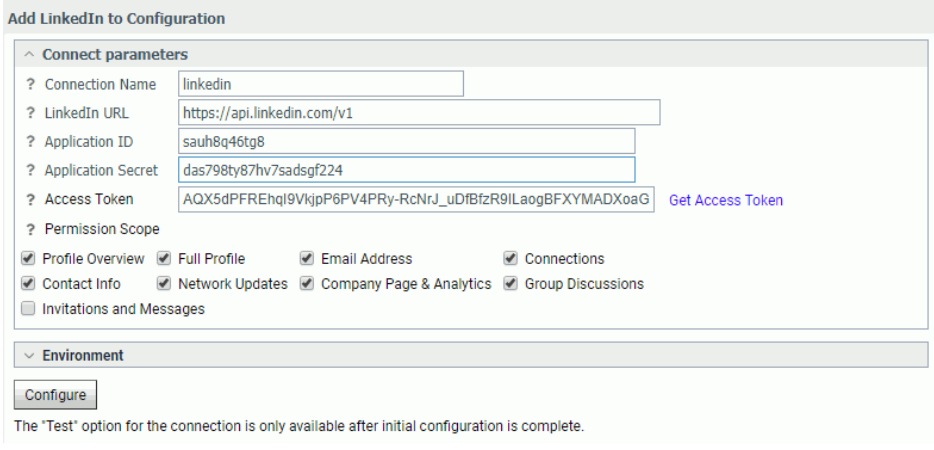
8. Choose the Permission Scope options to be granted to the LinkedIn application and click the *Get Access Token* link. For more information, see *Connection Attributes for LinkedIn* on page 1283.

   A LinkedIn Sign In page opens, as shown in the following image.



9. Enter the LinkedIn Sign In credentials and then click *Allow Access*.

You are returned to the Add LinkedIn to Configuration pane, where the Access Token field is now populated, as shown in the following image.



10. Click *Configure*.

    The LinkedIn adapter is added to the configured Adapters list in the navigation pane.

    Note: The Access Token expires after 60 days. To refresh the Access Token, click the *Get Access Token* link and then click *Configure*.

Reference: Connection Attributes for LinkedIn

The following list describes the connection attributes for the LinkedIn adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

LinkedIn URL

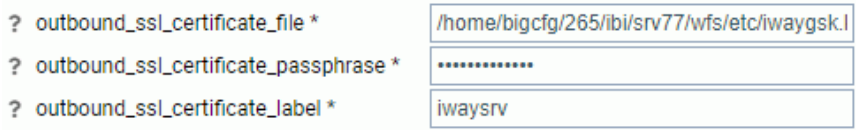
The URL of the LinkedIn API request. The default value is:

`https://api.linkedin.com/v1`

For iSeries machines, the WebFOCUS Reporting Server must be configured for SSL as follows:

1. From the Web Console sidebar, click *Workspace*.
2. From the menu bar, click *Workspace Set*, and select *Miscellaneous Settings*

3. Enter values for *outbound_ssl_certificate_file*, *outbound_ssl_certificate_passphrase*, and *outbound_ssl_certificate_label*, and then click *Save*. For example:



Application ID

The LinkedIn Application ID as defined in the LinkedIn application. For more information, see *Creating a LinkedIn Application* on page 1275.

Application Secret

The LinkedIn Application Secret as defined in the LinkedIn application. For more information, see *Creating a LinkedIn Application* on page 1275.

Access Token

Click the *Get Access Token* link to obtain this token. The credentials for a LinkedIn account are then entered. The value for the Access Token is returned by an authorized login.

Permission Scope

Grants the selected permissions to the LinkedIn application, which include:

- ❑ Profile Overview
- ❑ Full Profile
- ❑ Email Address
- ❑ Connections
- ❑ Contact Info
- ❑ Network Updates
- ❑ Company Page & Analytics
- ❑ Group Discussions
- ❑ Invitations and Messages

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the connection attributes. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), choose New Profile from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

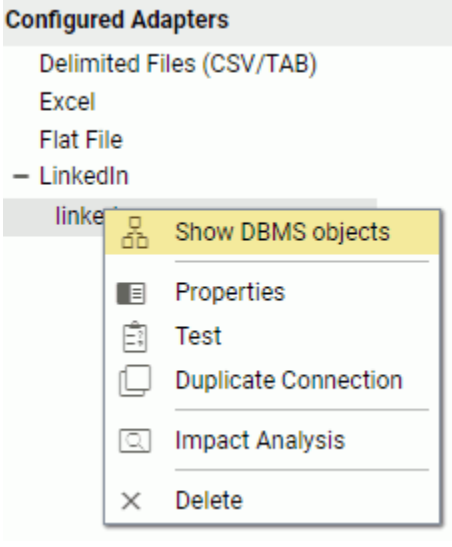
Store the connection attributes in the server profile (edasprof).

Creating Metadata and Sample Reports for the LinkedIn Adapter

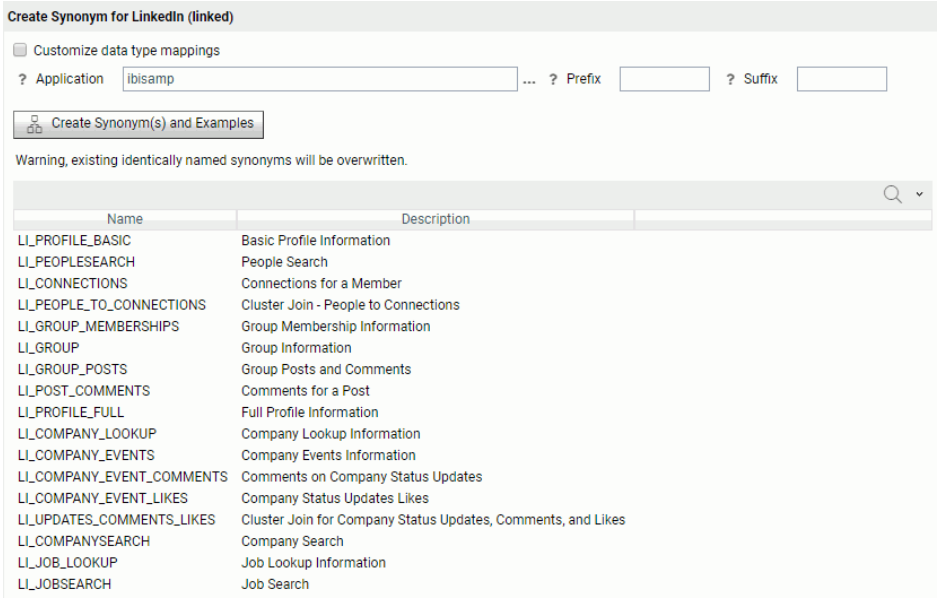
Create Synonym for the LinkedIn Adapter creates the metadata used for WebFOCUS reporting. It also creates sample WebFOCUS reports.

Procedure: How to Create Metadata and Sample Reports

1. From the WebFOCUS Reporting Server Web Console, expand the *Adapters* folder, *Configured* folder, and then the *LinkedIn* folder.
2. Right-click the configured connection for the LinkedIn Adapter (for example, linkedin) and select *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for LinkedIn pane opens, as shown in the following image.



3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata and sample reports are to be stored.
4. Click *Create Synonym(s) and Examples*.

   The Create Synonym for LinkedIn Status pane opens and indicates that the synonym was created successfully.

LinkedIn Adapter Examples

This section describes the metadata and sample reports for the LinkedIn Adapter.

Reference: LinkedIn Adapter Metadata

The following table lists and describes the available metadata for the LinkedIn Adapter.

Metadata	Description
li_company_event_comments	Used to retrieve company event status updates and comments.
li_company_event_likes	Used to retrieve company event status update likes.

Metadata	Description
li_company_events	Used to retrieve a company's status updates and job postings.
li_company_lookup	Used to lookup a company profile.
li_companysearch	Used to search for companies based on keywords.
li_connections	Used to retrieve connections for a LinkedIn member.
li_group	Used to retrieve information about a specific group.
li_group_memberships	Used to retrieve a list of groups that a LinkedIn member has joined.
li_group_posts	Used to retrieve posts and comments for a specific group.
li_job_lookup	Used to lookup details for a specific job.
li_jobsearch	Used to search for jobs based on search criteria.
li_peoplesearch	Used to search for people based on search criteria.
li_post_comments	Used to retrieve comments for a specific group post.
li_profile_basic	Used to retrieve basic profile information for a LinkedIn member.
li_profile_full	Used to retrieve full profile information for the authenticated LinkedIn member.
lisampl/li_people_to_connections	Cluster Join from li_profile_basic to li_connections.

Metadata	Description
lisampl/li_updates_comments_likes	Cluster Join to report on company status updates, comments, and likes. Joins: ❑ li_company_events to li_company_event_comments ❑ li_company_events to li_company_event_likes

Reference: **LinkedIn Adapter Sample Reports**

The following table lists and describes the sample reports for the LinkedIn Adapter.

Sample Report	Description
lisampl/ li_company_event_comments	Reports on company events including comments. Uses: li_company_event_comments
lisampl/li_company_event_likes	Reports on the member likes for company status updates. Uses: li_company_event_likes
lisampl/li_company_events	Reports on company status updates and job postings for a specific company. Uses: li_company_events
lisampl/li_company_lookup	Retrieve information about a specific company. Uses: li_company_lookup

Sample Report	Description
lisampl/li_companysearch	Searches for companies based on keywords. Uses: li_companysearch
lisampl/li_connections	Lists the connections for a specific LinkedIn member. Uses: li_connections
lisampl/li_group	Displays the information about a specific group. Uses: li_group
lisampl/li_group_memberships	Lists the groups that a LinkedIn member has joined. Uses: li_group_memberships
lisampl/li_group_posts	Reports on posts including comments for a specific group. Uses: li_group_posts
lisampl/li_job_lookup	Displays information about a specific job. Usage requires approval from LinkedIn to Vetted API Access where the application form can be accessed from: *https://help.linkedin.com/app/api-dvr* Uses: li_job_lookup

Sample Report	Description
lisampl/li_jobsearch	Searches for jobs based on specific search criteria. Usage requires approval from LinkedIn to Vetted API Access where the application form can be accessed from: *https://help.linkedin.com/app/api-dvr* Uses: li_jobsearch
lisampl/li_people_to_connections	Lists the connections for a specific LinkedIn member. Includes name of the specific LinkedIn member. Uses: lisampl/li_people_to_connections
lisampl/li_peoplesearch	Search for people based on specific search criteria. Usage requires approval from LinkedIn to Vetted API Access where the application form can be accessed from: *https://help.linkedin.com/app/api-dvr* Uses: li_peoplesearch
lisampl/li_peoplesearch_facet	Performs a facet search based on specific search criteria. For example, location and industry. Usage requires approval from LinkedIn to Vetted API Access where the application form can be accessed from: *https://help.linkedin.com/app/api-dvr* Uses: li_peoplesearch_facet

Sample Report	Description
lisampl/li_post_comments	Reports on comments for a specific group post. Uses: li_post_comments
lisampl/li_profile_basic	Displays the basic profile information for a specific LinkedIn member. Uses: li_profile_basic
lisampl/li_profile_full	Displays the full profile information for the authenticated LinkedIn member. Uses: li_profile_full
lisampl/li_updates_comments_likes	Displays the status updates, comments and likes for a specific company. Uses: lisampl/li_updates_comments_likes

Chapter 53

Using the Adapter for Lotus Notes

Based on the native client API, the Adapter for Lotus Notes provides access to local Lotus Notes databases, including email databases, and remote Domino Server databases.

The adapter currently creates synonyms for documents based on forms, views, folders, and calendars, which are referred to collectively as objects in the remainder of this document.

In this chapter:

Preparing the Lotus Notes Environment

The Adapter for Lotus Notes accesses Domino databases using Java APIs, which are part of the Domino Designer package.

The adapter can access both local Lotus Notes databases and remote Domino Server databases:

- If, on a local computer, Lotus Notes Client or Domino Designer or Domino Server is installed, then you must include the Notes.jar file in the CLASSPATH.
- If the local computer does not have any Lotus Notes software installed and you wish to access remote Domino Server databases through the adapter, you must include an NCSO.jar file in the CLASSPATH.
- On a remote Domino server, CORBA-based remote (IIOP) APIs calls are employed. The local computer need not have Domino installed.

For related information, refer to your *Lotus Notes* documentation.

Procedure: How to Prepare the Lotus Notes Environment on Windows

Specify the location of the Lotus Notes .jar file in the CLASSPATH environment variable.

Procedure: How to Prepare the Lotus Notes Environment on UNIX

Specify the location of the Lotus Notes .jar file in the $CLASSPATH environment variable.

For example, if the files are located in /usr/driver_files, you would issue the following statements:

```
CLASSPATH=/usr/driver_files/NCSO.jar
export CLASSPATH
```

Alternatively, you could use the Web Console to specify the location:

1. From the menu bar's Workspace menu, select *Configuration/Monitor*.
2. Expand the *Java Services* folder.
3. Right-click *DEFAULT* and select *Properties* from the popup menu.

   The Java Services Configuration pane opens.
4. Select the *Class path* section.
5. Specify full path to the Lotus Notes .jar file in the *IBI_CLASSPATH* text field.
6. Click the *Save and Restart Java Services* button.

Configuring the Adapter for Lotus Notes

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to Lotus Notes, the adapter requires connection information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Lotus Notes

The Lotus Notes adapter is under the *DBMS* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Note: For a local database connection, this is the only required entry.

Domino Server host

The remote server name.

Port

Domino Server port for remote (IIOP) calls.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Connection string for local database:

```
ENGINE LOTUS SET CONNECTION_ATTRIBUTES connection_name
```

Connection string for remote database:

```
ENGINE LOTUS SET CONNECTION_ATTRIBUTES connection_name
 Domino_server/user_ID,password: 'PORT port'
```

where:

connection_name

Is the logical name used to identify a particular set of attributes.

Note: For a local database connection, this is the only required entry.

Domino_server

Is the remote server name.

port

Is the Domino Server port for remote (IIOP) calls.

user_ID

Primary authorization ID by which you are known to Lotus Notes.

password

Password associated with the primary authorization ID.

Example: Declaring Connection Attributes

Connection string for local database:

```
ENGINE LOTUS SET CONNECTION_ATTRIBUTES CON2
```

Connection string for remote database:

```
ENGINE LOTUS SET CONNECTION_ATTRIBUTES CON1
ibilotus/SYSTEM,FEE21945C4B7942A:'PORT 63148'
```

Managing Lotus Notes Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Lotus Notes data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Lotus Notes object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

For the adapter to be able to create synonyms, Lotus Notes documents and data must be accessible to the HTTP client.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Web Console or the Data Management Console.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the synonym creation parameters reference.

   - *Synonym Creation Parameters for a Local Lotus Notes Database* on page 1299.
   - *Synonym Creation Parameters for a Remote Domino Database* on page 1302.
4. After entering the parameter values, click *Create Synonym*.

   The Status pane indicates that the synonym was created successfully.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the *Validate* check box, the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: **Synonym Creation Parameters for a Local Lotus Notes Database**

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Step 1

Select a database

Your previously specified connection information appears at the top of the pane. Notice that Domino Server host is indicated as <*local*>.

In the Local Database Full Name box, enter the name of the database you wish to use.

Specify Lotus Notes objects in the database

- By default object types are not restricted. If you wish to restrict them, deselect the corresponding check boxes.
- If you wish to retrieve all objects in the specified database, leave the *Filter by Name* box unchecked.
- If you wish to enter an object name or filter the objects list, check the *Filter by Name* box. A Name box opens in which you can specify a particular object or enter a string with wildcards as needed (for example, abc%, %abc, or %abc%) to limit the list of objects that will be displayed.

Complete Step 1 by clicking the *Next* button.

Step 2: Synonym field name processing options

Select options from this pane as needed.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Step 2 (continued): The objects retrieved from the database you selected in Step 1 are listed in the table at the bottom of the pane. You can now select objects to create synonym(s).

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Fixed Alpha Fields?

If this box is checked, all Lotus alpha fields are created as A*n*, where *n* is specified in the Alpha Field Size column.

If this box is unchecked, all Lotus alpha fields are created as A*n*V (the default), where *n* is specified in Alpha Field Size column.

Alpha Field Size

Displays the length of the alpha fields. The default length is 127. You can change this value. The generated Master File will reflect the chosen field length.

Database Name

Displays the name of the database that contains the selected objects.

Object Name

Lists the names of the objects for which you can generate synonyms. (The default synonym name is the same as the object name.)

Object Type

Indicates whether the object is a form, view, folder, or calendar.

Select Objects

Select objects for which you wish to create synonyms:

- To select all objects in the list, click the *Select All* button.
- To select specific objects, select the corresponding check boxes.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Example: Creating a Synonym for a Form in a Local Lotus Notes Database

From the Web Console or the Data Management Console, right-click the configured Lotus Notes connection and choose *Create Synonym* from the menu. The first synonym creation pane opens:

1. In the Local Database Full Name entry box, type: *log.nsf*
2. For this example it is not necessary to filter objects, so simply click *Next*.
3. On the second pane, select *Mail_Events* from the list of forms and views, and click *Overwrite Existing Synonyms*.
4. Leave the Fixed Alpha Fields check box unchecked (the default) to create all Lotus alpha fields in the format A*n*V.
5. The alpha fields length in the next column is 127 by default. Use the default value for this example.

6. Click the *Create Synonym* button.

   The synonym is created and added under the specified application directory (baseapp is the default).

   A status window displays the message: *Created successfully*

7. Click *Applications* on the menu bar.
8. Open the *baseapp* application folder in the navigation pane and click the synonym *Mail_Events*.
9. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File for Mail_Events form in the local log.nsf database

```
FILENAME=MAIL_EVENTS, SUFFIX=LOTUS    , $
  SEGMENT=MAIL_EVENTS, SEGTYPE=S0, $
    FIELDNAME=UNID, ALIAS=UNID_KEY, USAGE=A32, ACTUAL=A32, $
    FIELDNAME=BODY, ALIAS=Body, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
    FIELDNAME=FINISHTIME, ALIAS=FinishTime, USAGE=HYYMDs, ACTUAL=A17,
      MISSING=ON, $
    FIELDNAME=STARTTIME, ALIAS=StartTime, USAGE=HYYMDs, ACTUAL=A17,
      MISSING=ON, $
    FIELDNAME=SERVER, ALIAS=Server, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
```

Generated Access File

```
SEGNAME=MAIL_EVENTS, CONNECTION=CON3,DATABASE=log.nsf,OBJECT=Mail Events,
  OBJTYPE=FORM, KEYNAME=UNID_KEY, $
```

Reference: Synonym Creation Parameters for a Remote Domino Database

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Step 1

Specify databases

Your previously specified connection information appears at the top of the pane: Domino Server, port, and user ID. You can now specify the databases you want to retrieve from the remote Domino server.

- ❑ If you want to retrieve all databases, leave the box *Filter by Name* box unchecked.

- If you want to specify a database name or filter the list of retrieved databases, check the *Filter by Name* box. A Name box opens in which you can specify a particular database or enter one or more wildcards (for example, abc%, %abc, or %abc%) to limit the list of databases that will be displayed.

Complete Step 1 by clicking the *Next* button.

Step 2

Specify objects and the database that contains them

Your database filtering criterion appears at the top of the pane, and the list of databases that meet the criterion appears in a drop-down list at the bottom. In this step you specify a particular database and request retrieval of one, all, or a filtered list of the objects it contains.

- By default object types are not restricted. If you wish to restrict them, deselect the corresponding check boxes.
- If you wish to retrieve all objects in the selected database, leave the *Filter by Name* box unchecked.
- If you wish to specify an object name or filter the objects list, check the *Filter by Name* box. A Name box opens in which you can specify a particular object or enter one or more wildcards (for example, abc%, %abc, or %abc%) to limit the list of objects that will be displayed.
- If, in step 1, you requested a filtered list of databases or did not filter at all, you can now choose a specific database from the *Select Database* drop-down menu.

Complete Step 2 by clicking the *Next* button.

Synonym field name processing options (Step 3)

Select options from this pane, as needed.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

(Step 3 continued) The objects retrieved from the database you selected previously (either in step1 or step 2) are listed in the table at the bottom of the pane. You can now select objects to create synonym(s).

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Fixed Alpha Fields?

If this box is checked, all Lotus alpha fields are created as A*n*, where *n* is specified in the Alpha Field Size column.

If this box is unchecked, all Lotus alpha fields are created as A*n*V (the default), where *n* is specified in Alpha Field Size column.

Alpha Field Size

Displays the length of the alpha fields. The default length is 127. You can change this value when a synonym is created. The generated Master File will reflect the chosen field length.

Database Name

Displays the name of the database that contains the selected objects.

Object Name

Lists the names of the objects for which you can generate synonyms. (The default synonym name is the same as the object name.)

Object Type

Indicates whether the object is a form, view, folder, or calendar.

Select Objects

Select objects for which you wish to create synonyms:

- To select all objects in the list, click the *Select All* button.
- To select specific objects, select the corresponding check boxes.

Example: Creating a Synonym for a View in a Remote Domino Database

From the Web Console or the Data Management Console, right-click the configured Lotus Notes connection and choose Create Synonym from the menu. The first synonym creation pane opens:

1. Click the *Filter by Name* check box, type *certlog.nsf* in the Name entry box, and click the *Next* button. The second pane opens.
2. For this example, you will restrict objects to views, so leave only *Views* checked. Then click the *Next*button.
3. On the third pane, select the *By _User Name* view from the list of views, and click *Overwrite Existing Synonyms*.
4. Leave the Fixed Alpha Fields check box unchecked (the default) to create all Lotus alpha fields in the format A*n*V.
5. The alpha fields length in the next column is 127 by default. Use the default value for this example.
6. Click the *Create Synonym* button.

   The synonym is created and added under the specified application directory (baseapp is the default).

   A status window displays the message: Created successfully
7. Click *Applications* on the menu bar.
8. Open the *baseapp* application folder in the navigation pane and click the synonym *By_User_Name*.

9. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File for the By _User_Name view in the certlog.nsf database

```
FILENAME=BY_USER_NAME, SUFFIX=LOTUS    , $
  SEGMENT=BY_USER_NAME, SEGTYPE=S0, $
    FIELDNAME=USER_NAME, ALIAS=1, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
    FIELDNAME=LICENSE, ALIAS=2, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
    FIELDNAME=NOT_VALID_BEFORE, ALIAS=3, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
    FIELDNAME=CERTIFIER, ALIAS=4, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
    FIELDNAME=NOT_VALID_AFTER, ALIAS=5, USAGE=A127V, ACTUAL=A127V,
      MISSING=ON, $
```

Generated Access File

```
SEGNAME=BY__USER_NAME, CONNECTION=CON1, DATABASE=certlog.nsf,
  OBJECT=By_User Name, OBJTYPE=VIEW, $
```

Reference: Universal Note ID for Forms

A UNID field is automatically added to Master Files generated for Forms.

The Universal Note ID (UNID) identifies all copies of a note, regardless of location or time.

As a result, every replica of the note has the same UNID, which does not change when a note is modified.

(This field is illustrated in *Creating a Synonym for a Form in a Local Lotus Notes Database* on page 1301.)

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.

Keyword	Description
CONNECTION	Indicates access to the previously declared connection. The syntax is: CONNECTION=*connection*
DATABASE	Identifies the Lotus Notes database file.
OBJECT	Specifies the name of a Lotus Notes object.
OBJTYPE	Indicates the type of Lotus Notes object: FORM, VIEW, FOLDER, or CALENDAR.

Data Type Support

The following table indicates how the server maps Lotus Notes data types.

Lotus Notes Data Type	USAGE	ACTUAL	Remarks
Text Dialog List Checkbox Radio Button List Box Combo Box Authors Names Readers Password Formula	A*n*V | A*n*	A*n*V | A*n*	*n* is specified by user (default is 127).
Date/Time	HYYMDs	A17	
Number	D16	D8	

Lotus Notes Data Type	USAGE	ACTUAL	Remarks
Rich Text			BLOB (Binary Large Object) format is supported as a component of the Rich Text data type, which is used to distribute e-mail attachments. The attachment portion uses the BLOB format in the Master File. BLOB format is supported through the Reporting Server API. For an illustration, see *Mapping the Rich Text Data Type* on page 1308.

Example: **Mapping the Rich Text Data Type**

Rich text fields are used for large amounts of text or to embed or attach objects. For example, the Body field on a mail form is a rich text field that is mapped to three fields in the following Master File

```
FIELDNAME=BODY_ATTACHMENT, ALIAS=Body_ATTACHMENT, USAGE=BLOB, ACTUAL=BLOB,
  MISSING=ON, $
FIELDNAME=BODY_FNAME, ALIAS=Body_IZHBOPIA, USAGE=A256, ACTUAL=A256,
  MISSING=ON, $
FIELDNAME=BODY, ALIAS=Body, USAGE=A25V, ACTUAL=A25V,MISSING=ON, $
```

where:

`BODY_ATTACHMENT`

Is a BLOB field used to store the attachment. BLOB format is supported through the Reporting Server API.

`BODY_FNAME`

Is the file name of the original attachment.

`BODY`

Is the text in the field. For large amounts of text, you can manually modify the USAGE and ACTUAL lengths in the Master File.

Note: Formatted text, another common use of the rich text data type, is *not* currently supported by the adapter.

Chapter 54

Using the Adapter for LDAP

The Adapter for LDAP (Lightweight Directory Access Protocol) allows applications to access LDAP data sources. The adapter converts data or application requests into native LDAP statements and returns optimized answer sets to the requesting program.

This is a read-only adapter. Information Builders currently supports the following vendors: Sun, IBM, Novell, Microsoft, and OpenLDAP. Some vendors provide libraries that are available natively with the operating system and do not require installation. Some vendor libraries have to be installed.

Since the Adapter for LDAP operates at the protocol level, Information Builders recommends using the client that is native to the operating system. It is marked as preferred on the configuration page. There is no need to match the vendor of the LDAP server with the vendor of the LDAP client.

In this chapter:

Preparing the LDAP Environment

Before you can create an LDAP synonym, you must:

- **Install LDAP vendor's libraries.** For details, see *LDAP Vendor Library Prerequisites* in the Web Console help system (expand the *Server Administration* folder, then open the *Security* topic and choose *Security LDAP)*

Configuring the Adapter for LDAP

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In order to connect to LDAP, the adapter requires connection information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for LDAP

The LDAP adapter in under the *DBMS* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

LDAP Vendor

Name of the LDAP vendor: Sun, IBM, Novell, Microsoft.

LDAP Host

Address of the host machine on which the LDAP server is located.

Port

The LDAP server uses two different ports, one for regular connections (the default is 389); the other is for SSL connections (the default is 636).

To connect to LDAP through the SSL connection (secure connection), enter an LDAP port that is specifically configured by LDAP to accept SSL connections.

Secure Connection

This configuration parameter indicates an SSL/TLS connection to the LDAP server.

Security

There are two authentication methods when connecting to an LDAP server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the LDAP server at connection time for authentication.
- **Trusted.** The adapter connects to the LDAP server using anonymous bind.

User DN

User Distinguished Name, the unique identifier for an object in the LDAP Directory utilized as the primary authorization ID.

Password

Password associated with the primary authorization ID.

SSL Certificate

This option applies only if an ldap_secure_connection has been set to use a Secure Socket Layer (SSL) session with the LDAP server.

Enter the name of the LDAP attribute used by the API to establish the SSL/TLS connection. The API can be one of the following:

`Novell API, OpenLDAP API`

Specifies the file name (including path) of the Trusted Root Certificate that the LDAP server provides for authentication.

`Sun/Netscape API`

Specifies the path to cert7.db (Netscape certificate database excluding the file name) that the LDAP server provides for authentication.

`IBM API`

Specifies file name (including path) of the ldapkey.kdb (IBM key database file) that the LDAP server provides for authentication. (Note that the ldapkey.sth password stash file must be in the same directory.)

SSL Certificate Encryption

For Novell only, in the *ldap_ssl_certificate_encoding* field select the standard used to encode the certificate from the drop-down list. The options are:

`B64`

`DER`

Note that encryption and file format depend on API vendor specifications.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE X500IN SET CONNECTION_ATTRIBUTES conn_name ldap_server_url/
['user_dn'[,password]]:'PORT ldap_server_port_number
[SSLCERT ssl_certificate SSLCERTENC ssl_certificate_encoding]'
```

where:

`x500IN`

Is the suffix for the Adapter for LDAP.

conn_name

Is a logical name used to identify this particular set of attributes.

ldap_server_url

Is the address of the host machine on which the LDAP server is located.

user_dn

Is the User Distinguished Name, a unique identifier for an object in LDAP Directory utilized as the primary authorization ID.

password

Is the password associated with the primary authorization ID.

ldap_server_port_number

Is the LDAP server port number. (The SSL port should be used for the secure connection.)

ssl_certificate

Is the location of the SSL certificate file.

ssl_certificate_encoding

SSL certificate file encoding.

Note: UDNBASE and UDNATTRIBUTE are required for the Explicit security. Otherwise, they are ignored.

Example: Declaring Connection Attributes

```
ENGINE X500IN SET CONNECTION_ATTRIBUTES CON02 edasol29/
'uid=pgmavv,ou=iway,dc=ibi,dc=com',CD25FFDBB91B6790:'PORT 389'
```

Managing LDAP Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the LDAP data types.

Mapping Server Metadata and LDAP Schema Definitions

Server metadata is generated from the LDAP schema, which is stored on the LDAP server in a partition separate from the partitions that contain directory objects (which are treated as reported data objects).

Each directory object in the LDAP tree is addressable using a Distinguished Name (DN), which contains the root path from the object up.

The schema defines set of rules that govern the types of objects that can exist in a tree. Each object belongs to an object class that specifies which attributes can be associated with the object. All attributes are based on a set of attribute types that are, in turn, based on a standard set of attribute syntaxes. The schema controls the structure of individual objects as well as the relationships among the objects in the tree. Schema rules allow some objects to contain other, subordinate objects. Thus, the schema gives structure to the tree.

The schema consists of two basic components:

- **Object classes.** An Object class is a set of rules that determines what attributes can be contained in the directory object (entry).
- **Attribute types.** Attribute type is a set of data types called attribute syntaxes. The attribute syntaxes define the data types for values stored in the attribute.

Although LDAP can support many attribute syntaxes (data types), the Adapter for LDAP currently supports only the String data type.

Supported Object class rules are:

- **SUP.** Superior object class (parent).
- **MUST.** Required attributes (fields).
- **MAY.** Optional attributes (fields with MISSING=ON).

An LDAP Entry is the actual data item (object or node) that comprises the LDAP tree. Each entry holds Attributes, which are *key=value* pairs in which the key can have more than one value. Multi-value attributes are the default. Single-value attributes are denoted with the SINGLE-VALUE keyword in the attributeTypes attribute in the schema.

Server metadata describes a subset of the LDAP tree, starting with the DN passed down to the leaf hierarchical level when a synonym is created. The provided root DN is stored in the Access File.

The Server metadata represents each object in the processed LDAP hierarchy as a segment (whose name is taken from the Object class name), with fields (that are created out of the object attributes).

Master File	LDAP Schema
Segment name	Object class name
Field name	Adjusted Attribute name
Alias	Unadjusted Attribute name
Fields with MISSING=ON	Optional Attributes

The default USAGE and ACTUAL formats are set as A64 unless the size of an attribute is specified by the schema.

Creating Synonyms

Synonyms define unique names (or aliases) for each LDAP table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.

- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for LDAP

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Processing Modes are SCHEMA FLAT, SCHEMA HIERARCHY, and LDIF:

SCHEMA FLAT

Builds a synonym ignoring the LDAP data tree hierarchy, and produces a Master File with a single segment.

SCHEMA HIERARCHY

Builds a synonym reflecting the LDAP data tree hierarchy supporting two levels of the data tree (root and child), and produces a Master File by mapping selected object classes to segments.

LDIF

Builds a synonym reflecting the complete hierarchy of the LDAP data tree and produces a multisegmented Master File.

Filter by Object Class name

Creates a subset of Object Classes so that only a small list of Object Classes is produced.

Set as Index

Indicates the attribute (field) that can be utilized as the sort attribute for the LDAP API.

Note: Only a single attribute can be used as an Index.

View referenced Object names

Produces a list of Object Classes related by inheritance to the one selected.

Note: Adding referenced Object(s) is optional.

Base DN

- **For SCHEMA FLAT:** Is the Distinguished Name (DN) of the LDAP tree entry (node) that is set as a default root for data retrieval.

  Note: While Base DN is optional at the Create Synonym step, it is required for data retrieval, and can be provided in TABLE requests as:

  ```
  IF BASEDN EQ 'dc=ibi,dc=com'
  ```

  Note: Range retrieval is supported. When range retrieval is to be utilized, it is necessary to set Base DN to the leaf object containing attributes with the description utilized for retrieval. It is the DN of the group when retrieving a list of group members via range retrieval.

- **For LDIF:** Is the Distinguished Name (DN) of the LDAP tree entry (node) from which the synonym is created.

Synonym name

Displays the name that will be assigned to the synonyms. To assign a different name, replace the displayed value.

Model DN

Distinguished Name that the adapter uses to retrieve list of attributes. It can be any valid DN of the LDAP tree. For example:

```
uid=pgmtst5,ou=iway,dc=ibi,dc=com
```

or

```
ou=iway,dc=ibi,dc=com
```

By default, it is root DSE:

```
dc=ibi,dc=com
```

If this step skipped, all the attributes in the schema are fetched to the list.

Select attributes

Optional attribute selection that allows the user to specify the list of attributes (fields) that are present in the synonym. The user can reduce the size of the Master File by omitting unnecessary attributes.

If this step is skipped, all schema attributes will be present in the synonym.

Note that if all attributes are selected, the effect is the same as if no attributes are selected. For example, all available attributes in the schema will be present in the synonym.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Reference: Guidelines for Manually Editing an LDAP Master File

If you wish to delete non-essential information from a generated synonym, you can manually edit the Master File using the following editing guidelines:

- **Technique 1.** If you wish to remove individual fields from a Master File segment and other fields in that segment that are not essential to your work, remove the entire segment from the Master File.
- **Techniques 2.** If you wish to remove individual fields from a Master File segment but require other fields in that segment, note the following before editing the file:
  - You must preserve any fields referenced in the Access File in the format RDN=*fieldname*.

    The attribute name RDN is part of the group of attributes that comprise an LDAP entry. This field is part of the DN, and, as such, is always activated during data retrieval and must be included in the Master File.
  - As long as you retain the RDN attribute, you can delete other non-essential fields from the Master File.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Chapter 55

Using the Adapter for MariaDB

The Adapter for MariaDB allows applications to access MariaDB data sources. The adapter converts application requests into native MariaDB statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the MariaDB Environment

The Adapter for MariaDB requires the MariaDB Connector/J JDBC driver.

In order to report against Unicode data in MariaDB, you must enable Unicode support in the reporting server. For more information, see *MySQL and Unicode* on page 1635.

Procedure: How to Prepare the MariaDB Environment on Windows

1. Specify the location of the MariaDB Connector/JDBC driver files in the CLASSPATH environment variable. You must specify the full location and name of the jar file. For example, if the jar file is located in C:\Program Files\MariaDB\JDBC Driver, then specify:

   ```
   CLASSPATH= C:\Program Files\MariaDB\JDBC Driver/mariadb-java-
   client-1.5.3.jar.
   ```

2. Specify the location of the Javarun time environment or development kit in the JAVA_HOME or JDK_HOME environment variable. Either is acceptable.

   You must specify the location where the run time environment is installed. You should see a sub-directory bin in that directory. For example, if you have the run time environment in C:\Program Files\java\jre7 then specify:

   ```
   JAVA_HOME=C:\Program Files\java\jre7
   ```

   Alternately, if you have the full development kit installed in C:\Program Files\java \jdk1.7.0_05, then specify:

```
JDK_HOME= C:\Program Files\java\jdk1.7.0_05
```

3. Start (or restart) the server.

Note: You also need to restart the server if the driver has changed.

Procedure: How to Prepare the MariaDB Environment on UNIX

Specify the location of several files:

1. Specify the location of the MariaDB Connector/JDBC driver files in the $CLASSPATH environment variable.

   For example, if the files are located in /usr/driver_files, you would issue the following statements:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   To ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

   Alternatively, you could use the Web Console to specify the location:

   a. From the Workspace menu, select *Java Services*.
   b. Right-click *DEFAULT*, and select *Properties*.

      The Java Services Configuration pane opens.
   c. Select the *Class path* section.
   d. Specify the full path to the MariaDB Connector/J jar file in the IBI_CLASSPATH field.
   e. Click *Save and Restart Java Services*.

2. Specify the location of the Java Development Kit's installation directory in the $JDK_HOME environment variable.

   For example, if you want to set the location of the Java Development Kit to /usr/java, you would issue the following statements:

   ```
   JDK_HOME=/usr/java
   export JDK_HOME
   ```

3. Specify the location of the Java Virtual Machine's installation directory in the $LD_LIBRARY_PATH environment variable.

   For example, if you want to set the location of the JVM to /usr/j2sdk1.7.0_05/jre/lib/i386/server, you would issue the following statements:

   ```
   LD_LIBRARY_PATH=/usr/j2sdk1.7.0_05/jre/lib/i386/server
   export LD_LIBRARY_PATH
   ```

Note that if the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

4. Start (or restart) the server.

Note: You also need to restart the server if the driver has changed.

MariaDB and Unicode

The Adapter for MariaDB is implemented using JDBC. This implementation supports Unicode data stored in character fields with the CHARACTER SET set to UTF-8.

You must set the LANG environment variable in the edastart file or in a separate shell file before you start the server. For example, for American English you would export the following variable:

```
export LANG=EN_US.UTF-8
```

For details, see *Unicode Support* in the *Server Administration* manual.

Configuring the Adapter for MariaDB

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one MariaDB database server and to supply authentication attributes necessary to connect to the server.

You can declare connections to more than one MariaDB database server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Overriding the Default Connection* on page 1640). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The *first* SET CONNECTION_ATTRIBUTES command sets the default MariaDB database server to be used.
- If more than one SET CONNECTION_ATTRIBUTES command declares the same MariaDB database server, the authentication information is taken from the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for MariaDB

The MariaDB adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Enter the location URL for the MariaDB data source. The basic syntax is

```
jdbc:MariaDB://host:port/database
```

where:

host:*port*

Is the computer name or IP address and port on which the MariaDB database is located.

database

Is the name of the database.

These are two examples:

```
jdbc:MariaDB://localhost/qatst
```

```
jdbc:MariaDB://<host>:<port>/qatst
```

You can reference additional MariaDB connection properties in the URL. If you wish to do so, follow these guidelines: in the URL the first property must be preceded by the ? character, and the second and subsequent properties referenced in the URL must be preceded by the & character followed immediately by an | character, as illustrated in the following example.

Suppose that you wish to add the following connection properties:

```
sessionVariables=sql_mode=PIPES_AS_CONCAT
 zeroDateTimeBehavior=convertToNull
```

Enter the URL as follows:

```
jdbc:MariaDB://host/database?sessionVariables=sql_mode=P
IPES_AS_CONCAT&|
 zeroDateTimeBehavior=convertToNull
```

Note: The URL must be entered as a single line, without a space after the | character.

Driver name

Name of the Microsoft JDBC driver: `org.mariadb.jdbc.Driver`

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

For explicit authentication:

```
ENGINE SQLMAR SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

For password passthru authentication:

```
ENGINE SQLMAR SET CONNECTION_ATTRIBUTES connection/
```

where:

`SQLMAR`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

The name of the connection. You can give any name to the connection that you want.

`userid`

Is the primary authorization ID by which you are known to MariaDB.

`password`

Is the password associated with the primary authorization ID.

Example: **Declaring Connection Attributes**

The following SET CONNECTION_ATTRIBUTES command connects to the MariaDB database server named TEST with an explicit user ID and password:

```
ENGINE SQLMAR SET CONNECTION_ATTRIBUTES TEST/USERA,PWDA
```

The following SET CONNECTION_ATTRIBUTES command connects to the MariaDB database server named TEST using password passthru authentication:

```
ENGINE SQLMAR SET CONNECTION_ATTRIBUTES TEST/
```

Authenticating a User

There are two methods by which a user can be authenticated when connecting to a MariaDB database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Database or Password Passthru.** User ID and password received from the client application are passed to the MariaDB database server for authentication.

When a client connects to the server, the user ID and password are passed to MariaDB for authentication and are not authenticated by the server. To implement this type of authentication, start the server with security turned off. The server allows the client connection, and then stores an encrypted form of the client connection message to be used for connection to a MariaDB database server at anytime during the lifetime of the server agent.

Overriding the Default Connection

Once all MariaDB connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific MariaDB connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.
- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax: How to Select a Connection to Access

```
ENGINE SQLMAR SET DEFAULT_CONNECTION [connection]
```

where:

SQLMAR

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the last command will be the active connection name.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example: Selecting a Connection to Access

The following SET DEFAULT_CONNECTION command selects the MariaDB database server named TNSNAMEB as the default MariaDB database server:

```
ENGINE SQLMAR SET DEFAULT_CONNECTION datasource_name
```

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for the datasource_name.

Controlling Connection Scope

This topic explains how to set the scope of logical units of work using adapters. This is accomplished by the SET AUTODISCONNECT command.

A connection occurs at the first interaction with the declared database server.

Syntax: How to Control the Connection Scope

```
ENGINE SQLMAR SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLMAR

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing MariaDB Metadata

This topic describes how to use CREATE SYNONYM for MariaDB data sources. It also describes MariaDB data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each MariaDB table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for MariaDB

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Object name

Selecting this option adds the Object Name parameters to the screen.

Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *MySQL Data Type Support* on page 1648.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for MariaDB synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLMAR ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=TEST,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the MariaDB tablename. The tablename may include owner (schema) name. For example, TABLENAME=[*owner.*]*tablename*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them first in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	Set to Yes to enable autoincrementing.
START	Initial value in incrementing sequence
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

MariaDB Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Adapter for the MariaDB Environment

This topic describes how to set the adapter for the MariaDB environment.

PASSRECS

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Set PASSRECS

```
ENGINE SQLMAR SET PASSRECS {ON|OFF}
```

where:

`SQLMAR`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

MariaDB Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Chapter 56

Using the Adapter for Microsoft Access

The Adapter for Microsoft Access allows applications to access Microsoft Access data sources. The adapter converts application requests into native Microsoft Access statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Microsoft Access Environment

The Adapter for Microsoft Access minimally requires the installation of ODBC 32-bit Driver Manager. The configuration of a system data source name allows you to connect to a local or remote Microsoft Access data source.

Procedure: How to Set Up the Environment on Windows

There is no environment set up needed to access Microsoft Access.

Configuring the Adapter for Microsoft Access

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Microsoft Access database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Microsoft Access database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Microsoft Access Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Microsoft Access

The *MS Access* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

Password

The password, if any, associated with the data source.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Trusted authentication.

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES connection DSN_name
```

Explicit authentication. For password protected data sources:

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES connection DSN_name/,[password]
```

where:

`SQLMAC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

DSN_*name*

Is the Microsoft Access Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the Microsoft Access DSN named SAMPLESERVER with a password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

For example, suppose that prior 7.6.1 the connection was defined as:

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
```

When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

```
CONNECTION=DSN_A
```

If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

```
ENGINE SQLMAC SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLMAC SET DEFAULT_CONNECTION connection
```

where:

`SQLMAC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLMAC SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLMAC SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLMAC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Microsoft Access Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft Access data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft Access table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Microsoft Access

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Object name

Selecting this option adds the Object Name parameters to the screen.

Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings , see *Microsoft Access Data Type Support* on page 1352.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Microsoft Access synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=NF29004, SUFFIX=SQLMAC ,$
SEGNAME=NF29004, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I11, I4,  MISSING=ON ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I11, I4,  MISSING=ON ,$
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004,TABLENAME=NF29004,
CONNECTION=MACDSN,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Microsoft Access table name.
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Microsoft Access Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95

Enabling National Language Support

The SET parameter NCHAR indicates whether the character set is single-byte, double-byte, or triple-byte. The NCHAR setting affects the mapping of NCHAR and NVARCHAR data types.

The following chart lists data type mappings based on the value of NCHAR.

Microsoft Access Data Type	Remarks	NCHAR SBCS		NCHAR DBCS		NCHAR TBCS	
NCHAR (*n*)	*n* is an integer between 1 and 4000 *d* = 2 * *n* *t* = 3 * *n*	A*n*	A*n*	A*d*	A*d*	A*t*	A*t*
NVARCHAR (*n*)	*n* is an integer between 1 and 4000 *d* = 2 * *n* *t* = 3 * *n*	A*n*	A*n*	A*d*	A*d*	A*t*	A*t*

Syntax: **How to Enable National Language Support**

```
ENGINE SQLMAC SET NCHAR {SBCS|DBCS|TBCS}
```

where:

SQLMAC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

SBCS

Indicates a single-byte character set. SBCS is the default value.

DBCS

Indicates a double-byte character set.

TBCS

Indicates a triple-byte character set.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96

Chapter 57

Using the Adapter for Microsoft Azure SQL Data Warehouse

The Adapter for Microsoft® Azure SQL Data Warehouse allows applications to access cloud-based Azure SQL Data Warehouse data sources. The adapter converts data or application requests into native T-SQL statements and returns optimized answer sets to the requesting program.

In this chapter:

Preparing the Microsoft Azure SQL Data Warehouse Environment

Access to the Azure portal must be obtained from your Microsoft representative, and Azure SQL Data Warehouse must be created on this portal, if that was not already done.

The adapter supports access to Azure SQL Data Warehouse through two types of Microsoft clients:

- ODBC on Windows and Linux.
- JDBC everywhere.

Configuring the Adapter for Microsoft Azure SQL Data Warehouse

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Microsoft Azure SQL database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command.

You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user*.prf), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user*.prf), or in a group profile (if supported on your platform).

You can declare connections to more than one Microsoft Azure SQL databases by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Microsoft Azure SQL takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Microsoft Azure SQL Data Warehouse

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server

Name of the machine where Microsoft Azure SQL is running.

Security

There are three methods by which a user can be authenticated when connecting to a Microsoft Azure SQL:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to Microsoft Azure SQL, at connection time, for authentication as a standard login. This option requires that Azure SQL security be set to Azure SQL and Windows.
- **Password Passthru.** The user ID and password received from the client application are passed to Microsoft Azure SQL, at connection time, for authentication as a standard login. This option requires that Azure SQL security be set to Azure SQL and Windows.
- **Trusted.** The adapter connects to Microsoft Azure SQL as an operating system login using the credentials of the operating system user impersonated by the server data access agent. This option works with either of Azure SQL security settings

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Default Database

Name of the Microsoft Azure SQL database used for this connection. The database name, including path, must be enclosed in single quotation marks. This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.

Additional connection string keywords (optional)

Please refer to Microsoft Azure SQL documentation for all available connection string keywords used to change the behavior of the ODBC connection.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually on Windows

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft Azure SQL, at connection time, for authentication.

```
ENGINE SQLADW SET CONNECTION_ATTRIBUTES [connection]
 server/userid,password [;dbname]
[;additional_connection_string_keywords]
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft Azure SQL, at connection time, for authentication.

```
ENGINE SQLADW SET CONNECTION_ATTRIBUTES [connection]
 server/[;dbname][;additional_connection_string_keywords]
```

Trusted authentication. The adapter connects to Microsoft Azure SQL as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE SQLADW SET CONNECTION_ATTRIBUTES [connection]server/,[;dbname]
[;additional_connection_string_keywords]
```

Note: Enclose values that contain special characters in single quotation marks. If a value contains a single quotation mark, this quotation mark must be preceded by another single quotation mark, resulting in two single quotation marks in succession. For example, to specify the user ID Mary O'Brien, which contains both a blank and a single quotation mark, enter: 'Mary O' 'Brien'.

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLADW SET DEFAULT_CONNECTION connection
```

where:

`SQLADW`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLADW SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLADW SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLADW`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Microsoft Azure SQL Data Warehouse Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft Azure SQL Data Warehouse data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft Azure SQL Data Warehouse table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

The adapter supports the creation of metadata for certain types of MS SQL Native Synonyms. This feature is only available for Microsoft Azure SQL versions 2012 or higher.

You can create metadata for Native Synonyms under the following conditions:

- Native Synonyms must be created from a TABLE or VIEW base object.
- The base object (TABLE or VIEW) must exist on the MS Azure SQL targeted by the adapter connection string.
- Native Synonyms must be created with the base object described by a name consisting of not more than three components. Depending on the location and ownership of the base object within the targeted server, acceptable formats, are:

  ```
  object_name, schema_name.object_name,
  ```

  or

  ```
  database_name.schema_name.object_name.
  ```

The following types of Native Synonyms are not supported:

- Those based on stored procedures.
- Those based on objects existing on a linked MS Azure SQL.
- Those based on objects described with a 4-part name, such as:

  ```
  server_name.database_name.schema_name.object_name
  ```

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 1541.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Microsoft Azure SQL Data Warehouse

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Important: If you select Stored Procedures as your object type, the input parameters will be a little different from those described here. For details, refer to *Creating a Report Against a Stored Procedure* on page 1544.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check *Use current database* to use the database that has been set as the default database.
- Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

  Before selecting a database, if Use current database is checked, uncheck it.

To specify the intended database, choose from the Select database drop-down menu, which shows all databases on the targeted instance of Microsoft Azure SQL. Selecting Default Database will retain the database set during connection configuration. If Default Database was not set during configuration, the database assigned to the active login on the Azure SQL will be used as the default.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, run the Data Types report from the Help menu on the ribbon of the Adapters page of the Server Web Console.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Microsoft Azure SQL Data Warehouse synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLADW ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=edaqa.nf29004,
CONNECTION=connmss, KEYS=1, WRITE=YES,$
```

Reference: Mapping Microsoft SQL ODBC Table Comments Into a Synonym

When you generate a synonym for a Microsoft SQL table or view, the adapter maps comments as follows:

- MS Azure SQL table/view comments (if present) are mapped to the REMARKS attribute in the Master File synonym.
- MS Azure SQL column comments (if present) are mapped to the DESCRIPTION attribute in the Master File synonym.

Both Unicode and non-Unicode comments are supported.

Also, MS Azure SQL column title (if present) is mapped to the TITLE attribute in the Master File synonym.

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Microsoft Azure SQL table. The table name can be fully qualified as follows: TABLENAME=[[*database.*]*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.

Keyword	Description
KEY	Specifies the columns that participate in the primary key without having to describe them first in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❏ KEYFLD is the FIELDNAME of the common column from the parent table. ❏ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	When set to *Yes*, enables the auto increment feature.
START	Initial value in incrementing sequence.
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Microsoft Azure SQL Data Warehouse Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLADW SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Support of Read-Only Fields

CREATE SYNONYM creates a field description with FIELDTYPE=R for Microsoft Azure SQL columns created as TIMESTAMP or columns with the IDENTITY attribute. These fields are read-only. When executing a MAINTAIN or MODIFY procedure, the adapter suppresses all write operations against columns marked in the Master File with FIELDTYPE=R.

Example: Supporting a Read-Only Field

This example creates a table in which the first column has the IDENTITY property and the second column is a timestamp column:

```
CREATE TABLE TAB1
 (idproptab int IDENTITY (1,1), timstmp timestamp)
```

CREATE SYNONYM generates the following Master File for this table:

```
FILE=TAB1, SUFFIX=SQLADW ,$
SEGNAME=TAB1, SEGTYPE=S0 ,$
FIELD=IDPROPTAB, idproptab, I11, I4,  MISSING=OFF, FIELDTYPE=R ,$
FIELD=TIMSTMP,   timstmp,   A16, A16, MISSING=ON,  FIELDTYPE=R ,$
```

Reporting Against a Microsoft Azure SQL Data Warehouse Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Microsoft Azure SQL Data Warehouse stored procedures and report against the procedure output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters: OUT parameters, and INOUT parameters in OUT mode, as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 1541.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 1544.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

A synonym describes the stored procedure parameters and answer set.

An answer set structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, *input parameters* refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception. If you know the internal logic of the procedure, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

Example: Synonym for Microsoft Azure SQL Data Warehouse Stored Procedure CustOrders

The following synonym describes a Microsoft Azure SQL Data Warehouse stored procedure with one input parameter, one output parameter, and one answer set containing four variables.

The Master File synonym is:

```
FILENAME=CUSTORDERS, SUFFIX=SQLADW  , $
   SEGMENT=INPUT, SEGTYPE=S0, $
      FIELDNAME=@CUSTOMERID, ALIAS=P0001, USAGE=A5, ACTUAL=A5,
         MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
   SEGMENT=OUTPUT, SEGTYPE=S0, PARENT=INPUT, $
      FIELDNAME=@RETURN_VALUE, ALIAS=P0000, USAGE=I11, ACTUAL=I4, $
   SEGMENT=ANSWERSET1, SEGTYPE=S0, PARENT=INPUT, $
      FIELDNAME=ORDERID, ALIAS=OrderID, USAGE=I11, ACTUAL=I4, $
      FIELDNAME=ORDERDATE, ALIAS=OrderDate, USAGE=HYYMDs, ACTUAL=HYYMDs,
         MISSING=ON, $
      FIELDNAME=REQUIREDDATE, ALIAS=RequiredDate, USAGE=HYYMDs,
         ACTUAL=HYYMDs, MISSING=ON, $
      FIELDNAME=SHIPPEDDATE, ALIAS=ShippedDate, USAGE=HYYMDs,
         ACTUAL=HYYMDs, MISSING=ON, $
```

The Access File synonym is:

```
SEGNAME=INPUT, CONNECTION=ITarget, STPNAME=Northwind.dbo.CustOrders, $
SEGNAME=OUTPUT, STPRESORDER=0, $
SEGNAME=ANSWERSET1, STPRESORDER=1, $
```

Reference: **Synonym Creation Parameters for Stored Procedures**

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Select *Stored Procedures*.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check Use current database to use the database that has been set as the default database.
- Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

Before selecting a database, if Use current database is checked, uncheck it.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Select

Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.

Name

The name of the synonym, which defaults to the stored procedure name.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters.

If all procedures have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Microsoft SQL Server ODBC Data Type Support* on page 1539.

Values

Select the check box for *every* parameter displayed for the specified procedure.

Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines:

- Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided.
- Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym creation purposes.

If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.

Creating a Report Against a Stored Procedure

You can report against a stored procedure answer set using the same facilities you use to report against a database table:

- **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 1546.
- **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 1544.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

IF

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

WHERE

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Microsoft Azure SQL Data Warehouse Environment

The Adapter for Microsoft Azure SQL Data Warehouse provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft Azure SQL.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLADW SET TIMEOUT {nn|0}
```

where:

`SQLADW`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console sidebar choose *Workspace*. In the navigation pane, expand *Java Services*. Right-click *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click it, and select *Stop.*

Specifying the Login Wait Time

You can use the LOGINTIMEOUT command to specify the number of seconds the adapter will wait for a response from Microsoft Azure SQL at connect time.

Note: For compatibility with previous releases of the adapter, TIMEOUT is available as a synonym for LOGINTIMEOUT.

Syntax: How to Specify the Login Wait Time

```
ENGINE SQLADW SET LOGINTIMEOUT|TIMEOUT {nn|0}
```

where:

SQLADW

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is approximately 15 seconds.

0

Represents an infinite period to wait for login response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the sidebar, right-clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

The available parameters are:

```
ENGINE SQLADW SET PASSRECS {ON|OFF}
```

where:

SQLADW

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Microsoft Azure SQL Data Warehouse Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Optimizing Requests if a Virtual Field Contains Null Values

The SET OPTNOAGGR command provides finely-tuned control of adapter behavior for optimization. Users who for any reason wish to prevent passing aggregation to the RDBMS can use this command. An example of such a reason might be where NULL values occur in aggregated data with calculations. The SET OPTNOAGGR command causes the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is instead performed internally by the server while JOIN and SORT operations are handled by the RDBMS.

If any DEFINE field contains calculations with NULL fields, then such operations cannot be translated to SQL and passed to DBMS because they always return NULL. It has to be processed by FOCUS.

This can be achieved by SET OPTIMIZATION OFF.

However, in some cases it is preferable to use the off-load JOIN and SORT operation to DBMS for better performance while leaving AGGREGATION to FOCUS.

Syntax: How to Set Enhanced Aggregation Control

```
SQL SQLADW SET OPT {AGGR|NOAGGR}
```

where:

AGGR

Directs the adapter to off-load aggregated DEFINE fields to the DBMS. This is the default setting.

```
NOAGGR
```

Directs the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is, instead, performed internally by the server, while JOIN and SORT operations are handled by the RDBMS. This setting can also be used to provide backwards compatibility for applications that were written based on the functionality of the previous release, when less SQL was off-loaded to the RDBMS. For example, when a calculation on aggregated fields may have contained NULL data that was not processed by the RDBMS NVL( ) function.

Example: Using IF-THEN_ELSE Optimization With a Condition That Is Always False

```
SQL SQLADW SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions, such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting. Consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

The Adapter for Microsoft Azure SQL Data Warehouse supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the sidebar, right-clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE SQLADW SET FETCHSIZE n
```

where:

SQLADW

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE SQLADW SET INSERTSIZE n
```

where:

SQLADW

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE SQLADW SET BULKLOAD [ON|OFF]
```

where:

SQLADW

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Uses the Bulk Insert API. ON is the default.

OFF

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Microsoft Azure SQL Data Warehouse.

For the Adapter for Microsoft Azure SQL Data Warehouse, the Bulk API is used automatically in LOADONLY mode. Measurements show that intermediate flushes do not affect performance; therefore, the behavior does not depend on the INSERTSIZE.

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Optimizing Non-Equality WHERE-Based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

LEFT_OUTER

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

hostfile

Is the host Master File.

AT

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

hfld1

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

tag1

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

WITH *hfld2*

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

MULTIPLE

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

UNIQUE

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations. It always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: Conditions for WHERE-Based Outer Join Optimization

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left Outer Join

The following request creates a left outer conditional join between two MSSQL data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE employee AT EMPLOYEEID
TO ALL FILE employeepayhistory AT EMPLOYEEID
     WHERE RATECHANGEDATE GT HIREDATE;
END
TABLE FILE employee
PRINT RATE
BY EMPLOYEEID
END
```

The WebFOCUS request is translated to a single MSSQL SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."EmployeeID", T1."HireDate", T2."EmployeeID",
T2."RateChangeDate", T2."Rate" FROM
AdventureWorks.HumanResources.Employee T1 LEFT OUTER JOIN
AdventureWorks.HumanResources.EmployeePayHistory T2 ON
(T2."RateChangeDate" > T1."HireDate") ) ORDER BY T1."EmployeeID";
```

Calling a Microsoft Azure SQL Data Warehouse Stored Procedure Using SQL Passthru

Microsoft Azure SQL stored procedures are supported using SQL Passthru. These procedures need to be developed within Microsoft Azure SQL using the CREATE PROCEDURE command.

The adapter supports stored procedures with input, output, and in-out parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets are returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to the rules of the underlying DBMS. Note that the examples shown in this section are SQL-based. See the DBMS documentation for rules, languages, and additional programming examples.

Syntax: How to Invoke a Stored Procedure

```
SQL SQLADW EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

SQLADW

Is the ENGINE suffix for Microsoft Azure SQL.

procname

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

You can employ either SQL or SYS naming conventions to control the separator character used for interpreting multipart names, as described in Setting Naming Conventions.

parameter_specification

IN, OUT, and INOUT parameters are supported. Use the variation required by the stored procedure:

IN

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

OUT

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

INOUT

Consists of a question mark (?) for output and a literal for input, separated by a slash (/). (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

In this example, a user invokes a stored procedure, edaqa.test_proc01, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL SQLADW EX edaqa.test_proc01 125,?,?/3.14,,'abc',,'xyz'
END
```

Example: Sample Stored Procedure

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (   OUT chSQLSTATE_OUT   CHAR(5),
                                  OUT intSQLCODE_OUT   INT,
                                  INOUT l_name char(20),
                                  INOUT f_name char(20))
    RESULT SETS 1
    LANGUAGE SQL
-------------------------------------------------------------------------
-- SQL Stored Procedure
-------------------------------------------------------------------------
P1: BEGIN
    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT
           EDAQA.NF29005.SSN5 AS SSN5,
           EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
           EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
           EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
           EDAQA.NF29005.SEX5 AS SEX5
        FROM
           EDAQA.NF29005
        WHERE
           (
             ( EDAQA.NF29005.LAST_NAME5 =  l_name )
        AND
             ( EDAQA.NF29005.FIRST_NAME5 = f_name )
           );
     -- Cursor left open for client application
    OPEN cursor1;
    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';
END P1   @
```

Reference: Capturing Application Errors in Stored Procedures

You can capture application errors using the RAISERROR method. Any application error that is issued by the stored procedure is available in the server variable &ADWMSGTXT.

Chapter 58

Using the Adapter for Microsoft Dynamics CRM

The Microsoft Dynamics CRM Adapter is used to report against the information residing in the Microsoft Dynamics CRM Online environment. For example, account, contact, campaign, and invoice information can be analyzed.

You can configure the Microsoft Dynamics CRM Adapter using the WebFOCUS Reporting Server Web Console. The adapter requires a connection that stores the refresh token. A valid Microsoft Dynamics CRM Online refresh token is required to issue Microsoft Dynamics CRM API calls. This token is associated with an App stored in the Microsoft Azure Active Directory and a specific Microsoft account that has access to the Microsoft Dynamics CRM Online environment, as well as the Microsoft Azure Active Directory.

In this chapter:

Creating an App in the Microsoft Azure Active Directory

An App must first be registered in the Microsoft Azure Active Directory before you can configure the Microsoft Dynamics CRM Adapter.

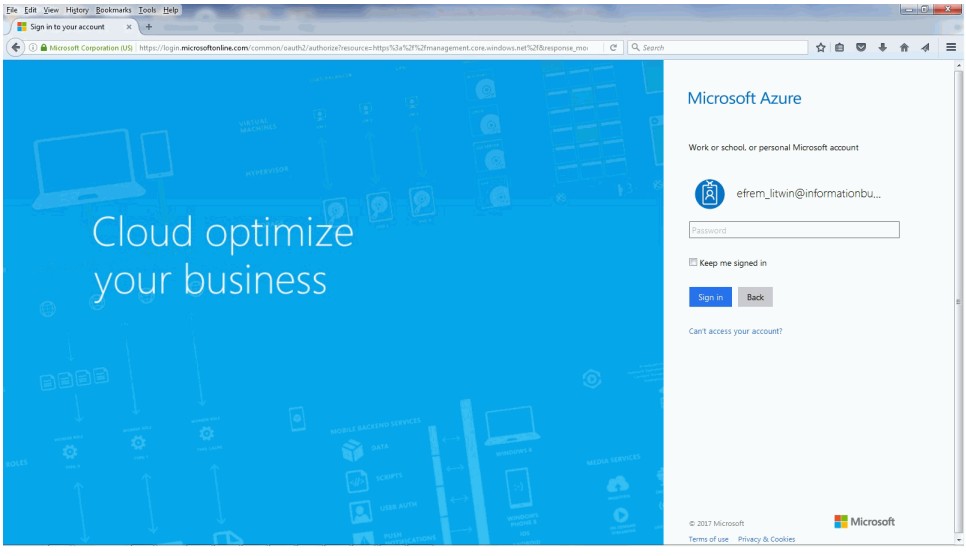
Procedure: How to Create an App in the Microsoft Azure Active Directory

The Microsoft account Administrator needs to ensure that the Microsoft account used in the Adapter configuration has access to the Microsoft Azure Active Directory.

1. Enter the following URL in a web browser.

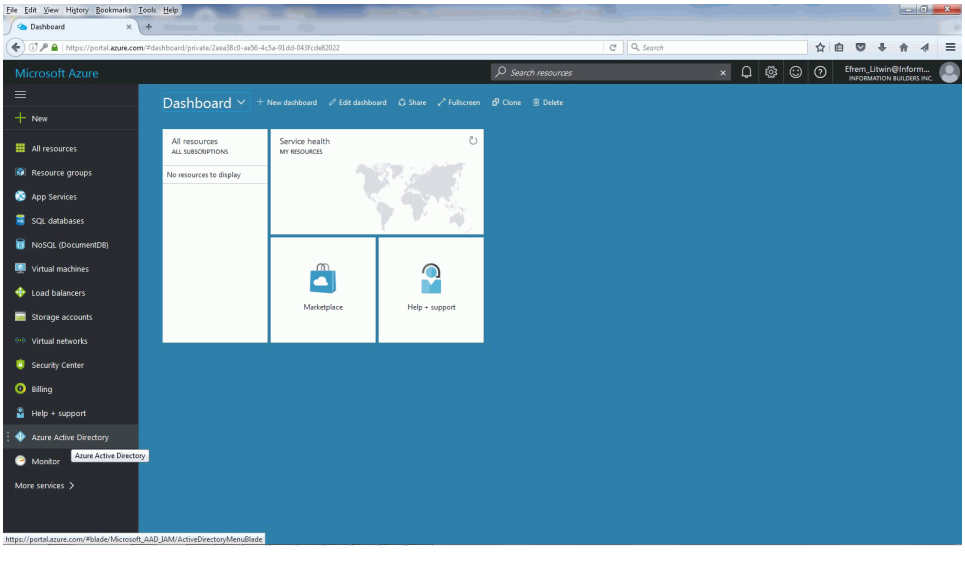
   https://portal.azure.com/

If you are not already signed into Microsoft Azure, a sign-in dialog for Microsoft Azure opens, as shown in the following image.



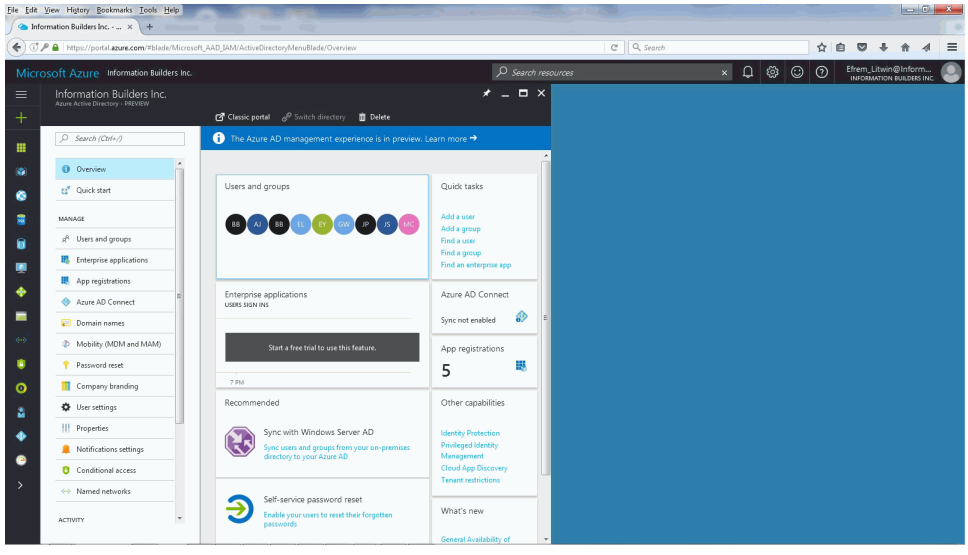
2. Enter valid Microsoft account credentials that have access to the Microsoft Azure Active Directory, and click *Sign in*.

   The Microsoft Azure Dashboard screen opens, as shown in the following image.



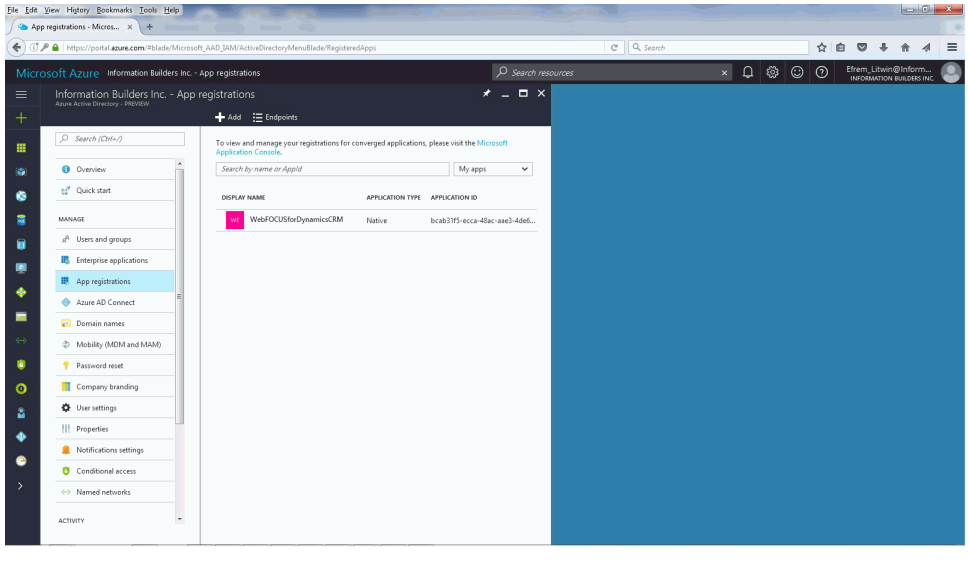
3. In the left panel, click *Azure Active Directory*.

The Azure Active Directory screen opens, as shown in the following image.



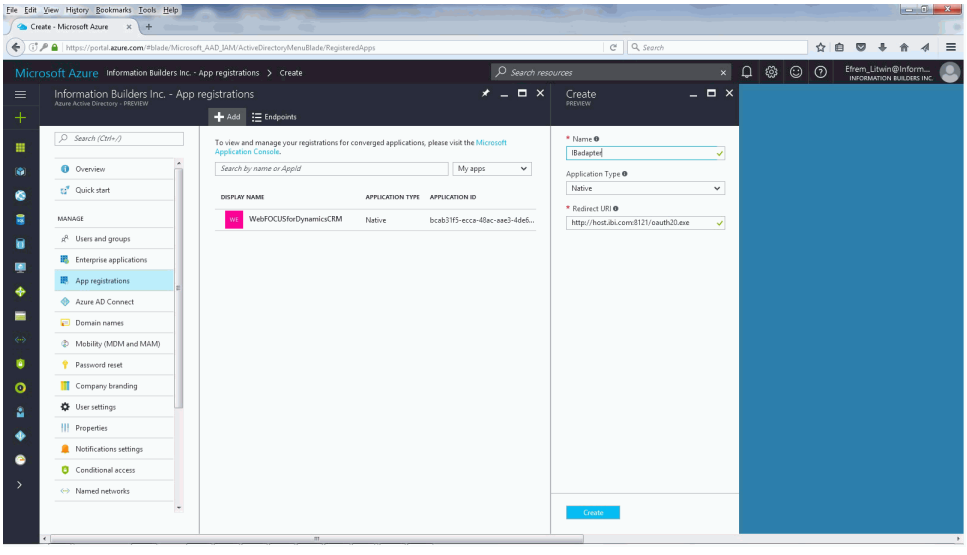
4. In the left panel, click *App registrations*.

   The App registrations screen opens, as shown in the following image.



5. Click + *Add*.

The Create panel opens, as shown in the following image.



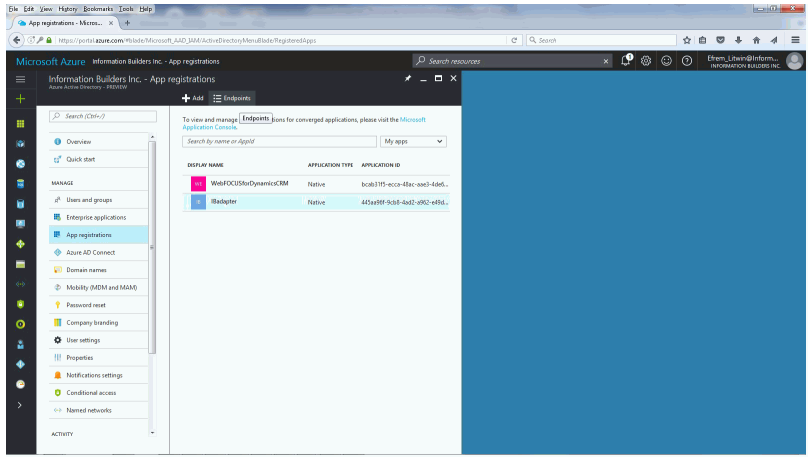
6. Enter the following values.
   a. Enter a Name for your new App.
   b. Select *Native* for the Application Type.
   c. Enter the host name and port used to access the WebFOCUS Reporting Server Web Console, appended with oauth20.exe, in the Redirect URI field.

      For example:

      `http://host.ibi.com:8121/oauth20.exe`

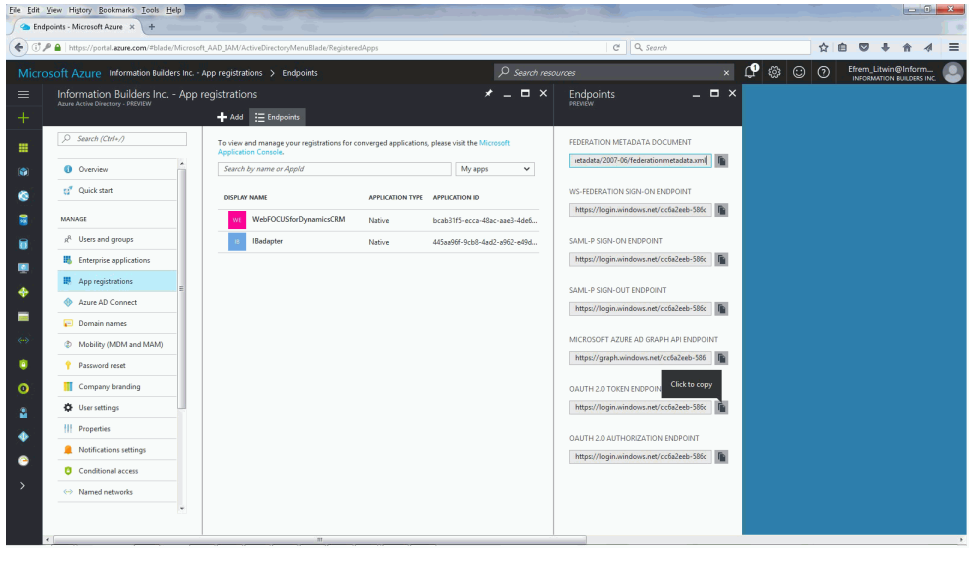
   d. Click *Create*.

The App is added to the App list, as shown in the following image



7. Click *Endpoints*.

   The Endpoints panel opens, as shown in the following image.



8. Click the *Copy* icon next to the OAUTH 2.0 TOKEN ENDPOINT field and paste the copied Endpoint to a document that can be referenced when configuring the Microsoft Dynamics CRM Adapter Connection.

   For example:

```
https://login.windows.net/cc6a2eeb-586d-4063-9057-xxxxxxxxxx/oauth2/token
```

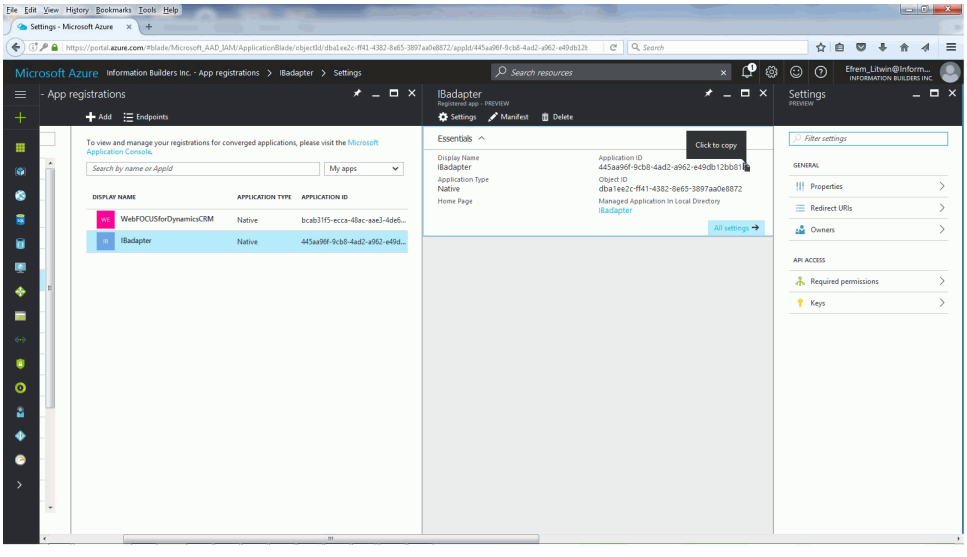
9. Click the *Copy* icon next to OAUTH 2.0 AUTHORIZATION ENDPOINT field and paste the copied Endpoint to a document that can be referenced when configuring the Microsoft Dynamics CRM Adapter Connection.

   For example:

```
https://login.windows.net/cc6a2eeb-586d-4063-9057-xxxxxxxxxx/oauth2/authorize
```

10. Click the App created in Step 6 from the center panel.

    The Essentials panel opens for the App, as shown in the following image.



11. Click the *Copy* icon next to the Application ID field and paste the copied Application ID to a document that can be referenced when configuring the Microsoft Dynamics CRM Adapter Connection.

    For example:

```
445aa96f-9cb8-4ad2-a962-e49db12bb81a
```

Configuring the Adapter for Microsoft Dynamics CRM

This section describes how to configure the Adapter for Microsoft Dynamics CRM.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

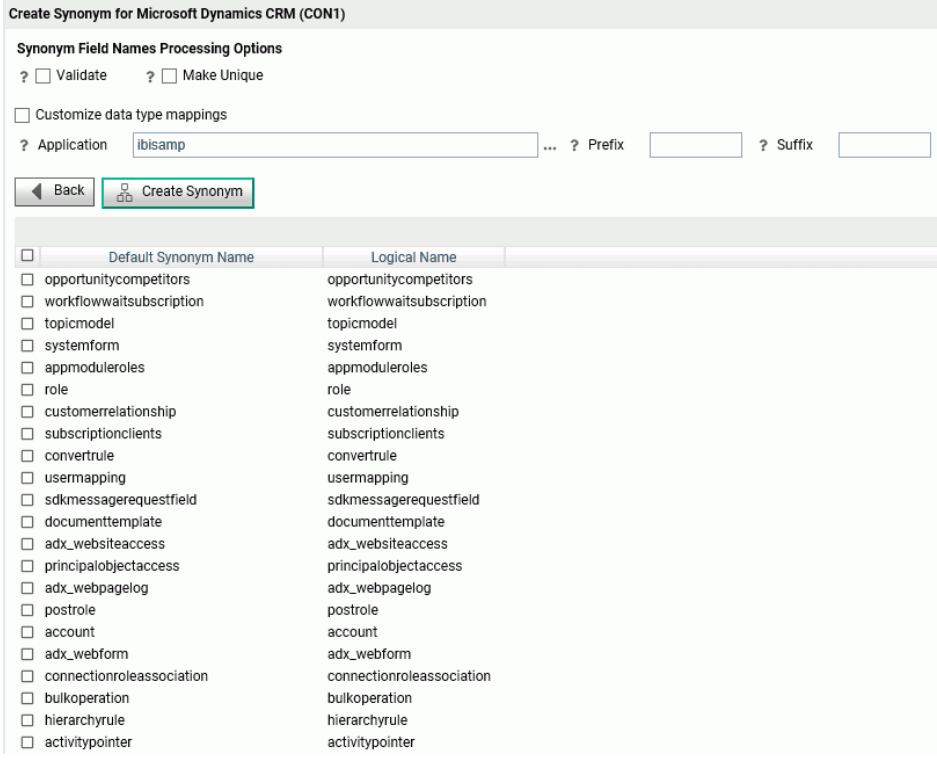
Creating Metadata for the Adapter for Microsoft Dynamics CRM

Create Synonym for the Microsoft Dynamics CRM Adapter creates the metadata used for WebFOCUS reporting against data returned from the Microsoft Dynamics CRM API calls.

Procedure: How to Create Metadata for the Adapter for Microsoft Dynamics CRM

1. From the Web Console *Applications* page, click *Get Data*.
2. Right-click the configured connection for the Adapter for Microsoft Dynamics CRM (for example, CON01) and click *Create Synonym* from the context menu, as shown in the following image.

The Create Synonym for Microsoft Dynamics CRM pane opens, as shown in the following image.



3. Enter a specific application in the Application field, or click the ellipsis button to the right of the field to select an application in which to store the metadata.

   a. Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.)

      This parameter ensures that names adhere to specifications. For more information, see *Validation for Special Characters and Reserved Words* on page 2743. When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

   b. Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

c. Click *Customize data type mappings* to select whether to decompose date formats, assign geographic roles automatically, set the date order, and select the data type mappings for numeric decimal and alphanumeric columns.

d. Select the check box next to the one or more synonyms for which metadata is to be created.

4. Click *Create Synonym*.

   The application where the synonyms were created opens.

Chapter 59

Using the Adapter for Microsoft SQL Server

The Adapter for Microsoft® SQL Server® allows applications to access Microsoft SQL Server data sources. The adapter converts data or application requests into native Microsoft SQL Server statements and returns optimized answer sets to the requesting program.

In this chapter:

Preparing the Microsoft SQL Server Environment

In order to take full advantage of the features available through the Adapter for Microsoft SQL Server 2012/2014, we strongly recommend using the same or higher version of the Microsoft SQL Server Client. To determine which version of MS SQL Server you are using, please refer to the following article: *http://support.microsoft.com/kb/321185*.

You can set up the Microsoft SQL Server Environment on Windows and UNIX. On Windows, for Microsoft SQL Server versions 2017/2016/2014/2012, you can choose between the ODBC version of the adapter (please refer to *Using the Adapter for Microsoft SQL Server ODBC* on page 1525) or the OLE DB version of the adapter (described in this chapter), depending on required features. Microsoft had deprecated the OLE DB version of Microsoft SQL Server, but has now un-deprecated it with the release of a third generation driver, MSOLEDBSQL driver version 18. To use the OLE DB version of the adapter for these releases, you must download and install MSOLEDBSQL driver version 18. Instructions for downloading and installing the driver can be accessed from the configuration page for the adapter by clicking the help link titled *Prerequisites*.

For version 2008, use the OLE DB version of the adapter (described in this chapter).

Procedure: How to Set Up the Environment on Windows (OLE DB)

The Microsoft SQL Server environment is set up during the installation of the Microsoft SQL Server, Client, and MDAC software.

No additional setup steps are required.

Procedure: How to Set Up the Environment on UNIX

Identify the location of the Microsoft SQL Server JDBC Driver files using the environment variable $CLASSPATH. For example, to set the location of the JDBC Driver files, specify:

```
CLASSPATH=/qas/mss/sqljdbc_4.0/enu/sqljdbc4.jar
export CLASSPATH
```

The driver sqljdbc4.jar file is within the "Microsoft JDBC Driver 4.0 for SQL Server" downloadable *.tar.gz version of the Microsoft package. Optionally, the CLASSPATH or IBI_CLASSPATH may be set up as part of the adapter configure step or in the Java Services properties (and exporting skipped, but the directory location must be known).

Identify the installation directory of the Java Development Kit using the environment variable $JDK_HOME. Java version 1.6 or higher is required. For example, if you want to set the location of the Java Development Kit to /usr/java, specify:

```
JDK_HOME=/usr/java
export JDK_HOME
```

Identify the installation directory of the Java Virtual Machine using the environment variable $LD_LIBRARY_PATH. For example, if you want to set the location of the Java Virtual Machine, specify:

```
LD_LIBRARY_PATH=/usr/java/jdk1.6.0_38/jre/lib/amd64/server
export LD_LIBRARY_PATH
```

Note: If the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

Procedure: How to Set Up the Environment on IBM i

Identify the location of the Microsoft SQL Server JDBC Driver files using the environment variable $CLASSPATH. For example, to set the location of the JDBC Driver files, specify:

```
CLASSPATH=/qas/mss/sqljdbc_4.0/enu/sqljdbc4.jar
export CLASSPATH
```

The driver file sqljdbc4.jar file is within the "Microsoft JDBC Driver 4.0 for SQL Server" downloadable *.tar.gz version of the Microsoft package. Optionally, the CLASSPATH or IBI_CLASSPATH may be set up as part of the adapter configure step or in Java Services property (and exporting skipped, but the directory location must be known).

JVM access is built-in on IBM i and no further environment set up is needed. Java version 1.6 or higher is required.

Accessing Microsoft SQL Server Remotely

You can access Microsoft SQL Server on a remote node. To access Microsoft SQL Server remotely, you must:

- Locally install the latest version of Microsoft SQL Server Native Client.
- Know the name of the remote Microsoft SQL Server instance.

All Microsoft SQL Servers installed are defined by using a unique NetBEUI name. The server can access any Microsoft SQL Server on the network, provided you define a valid user ID and password, as well as the name of the Microsoft SQL Server instance. You can define these parameters in either the server global profile or a user profile.

XA Support

Read/write applications accessing Microsoft SQL Server data sources are able to execute transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web Console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see *XA Support* on page 2689.

Configuring the Adapter for Microsoft SQL Server

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Microsoft SQL Server database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command.

You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Microsoft SQL Server databases by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Microsoft SQL Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.

On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Microsoft SQL Server

The *MS SQL Server* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server (Windows only)

Name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of Microsoft SQL Server installed, provide the server name and the instance name as follows: server\instance.

The server connection attribute will allow users to choose from the list of MS SQL Servers visible within the local network using the SQL Native Client Enumerator.

Please note that due to limitations of the SQL Native Client Enumerator, local network settings and MS SQL Server settings, it is possible that not all operational servers will be visible. If the name of the server you wish to target does not appear, you can enter it manually.

URL (UNIX, IBM i, and z/OS only)

Enter the location URL for the Microsoft SQL Server data source.

Security

There are three methods by which a user can be authenticated when connecting to a Microsoft SQL Server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication as a standard login.

  This option requires that SQL Server security be set to SQL Server and Windows (for Windows), or else to SQL Server and UNIX.

- **Password Passthru.** (*Windows only*) The user ID and password received from the client application are passed to Microsoft SQL Server, at connection time, for authentication as a standard login.

  This option requires that SQL Server security be set to: SQL Server and Windows.

- **Trusted.** The adapter connects to Microsoft SQL Server as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

  This option works with either of the SQL Server security settings.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Default Database (Windows only)

Name of the default database for the connection. This value is used when a data object is not qualified with the database name.

This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.

Additional connection string keywords (Windows only)

Any additional connection string keywords documented in the Microsoft OLE DB Driver for SQL Server documentation. This parameter is optional.

Driver name (UNIX, IBM i, and z/OS only)

Name for the Microsoft JDBC driver.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually on Windows

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES [connection]
server/userid,password [;dbname][:provider_string]
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES [connection]
server/[;dbname][:provider_string]
```

Trusted authentication. The adapter connects to Microsoft SQL Server as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES [connection]
server/,[;dbname][:provider_string]
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name (or a data source name) used to identify this particular set of attributes.

If you plan to have only one connection to Microsoft SQL Server, this parameter is optional. If not specified, the local database server serves as the default connection.

server

Is the name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of Microsoft SQL Server installed, provide the server name and instance as follows: server\instance.

When specifying the server name and the instance name, we recommend that you enclose the value in single quotation marks (').

userid

Is the primary authorization ID by which you are known to Microsoft SQL Server.

password

Is the password associated with the primary authorization ID.

dbname

Is the name of the Microsoft SQL Server database used for this connection. The database name, including path, must be enclosed in single quotation marks.

provider_string

Is the Microsoft SQL Server provider string used to specify additional connection options, such as the name of the network library. Note that this parameter must be preceded by a colon and enclosed in single quotation marks. This parameter is optional.

Note: Enclose values that contain special characters in single quotation marks. If a value contains a single quotation mark, this quotation mark must be preceded by another single quotation mark, resulting in two single quotation marks in succession. For example, to specify the user ID Mary O'Brien, which contains both a blank and a single quotation mark, enter: 'Mary O' 'Brien'.

Example: Declaring Connection Attributes on Windows

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Microsoft SQL Server database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Microsoft SQL Server database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Microsoft SQL Server database server using operating system authentication:

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES /,
```

Syntax: How to Declare Connection Attributes Manually on UNIX and z/OS

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES CON1 'URL'/userid,password
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`CON1`

Is the connection name.

`URL`

Is the URL to the location of the data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes on UNIX and z/OS

The following SET CONNECTION_ATTRIBUTES command specifies using the Microsoft JDBC Driver.

```
ENGINE SQLMSS SET JDBCDRIVERNAME
  com.microsoft.jdbc.sqlserver.SQLServerDriver
```

The following SET CONNECTION_ATTRIBUTES command connects to a myServer using the Microsoft SQL Server JDBC Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLMSS SET CONNECTION_ATTRIBUTES CON1
  'jdbc:microsoft:sqlserver://myServer:1433'MYUSER,PASS
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLMSS SET DEFAULT_CONNECTION connection
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLMSS SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLMSS SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Microsoft SQL Server Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft SQL Server table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

The adapter supports the creation of metadata for certain types of MS SQL Native Synonyms. This feature is only available for Microsoft SQL Server versions 2005 and higher.

You can create metadata for Native Synonyms under the following conditions:

- Native Synonyms must be created from a TABLE or VIEW base object.
- The base object (TABLE or VIEW) must exist on the MS SQL Server targeted by the adapter connection string.
- Native Synonyms must be created with the base object described by a name consisting of not more than three components. Depending on the location and ownership of the base object within the targeted server, acceptable formats, are:

  ```
  object_name, schema_name.object_name,
  ```

  or

  ```
  database_name.schema_name.object_name.
  ```

The following types of Native Synonyms are not supported:

- Those based on stored procedures.
- Those based on objects existing on a linked MS SQL server.
- Those based on objects described with a 4-part name, such as:

  ```
  server_name.database_name.schema_name.object_name
  ```

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 1421.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Microsoft SQL Server

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Important: If you select Stored Procedures as your object type, the input parameters will be a little different from those described here. For details, refer to *Creating a Report Against a Stored Procedure* on page 1425.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check Use current database to use the database that has been set as the default database.
- Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

  Before selecting a database, if Use current database is checked, uncheck it.

To specify the intended database, choose from the Select database drop-down menu, which shows all databases on the targeted instance of Microsoft SQL Server. Selecting Default Database will retain the database set during connection configuration. If Default Database was not set during configuration, the database assigned to the active login on the SQL Server will be used as the default.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Microsoft SQL Server Data Type Support* on page 1418.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Microsoft SQL Server synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLMSS ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=edaqa.nf29004,
CONNECTION=connmss, KEYS=1, WRITE=YES,$
```

Reference: Mapping Microsoft SQL Table Comments Into a Synonym

When you generate a synonym for a Microsoft SQL table or view, the adapter maps comments as follows:

- MS SQL Server table/view comments (if present) are mapped to the REMARKS attribute in the Master File synonym.
- MS SQL Server column comments (if present) are mapped to the DESCRIPTION attribute in the Master File synonym.

Both Unicode and non-Unicode comments are supported.

Also, MS SQL Server column title (if present) is mapped to the TITLE attribute in the Master File synonym.

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Microsoft SQL Server table. The table name can be fully qualified as follows: TABLENAME=[[*database*.]*owner*.]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them first in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	When set to *Yes*, enables the auto increment feature.
START	Initial value in incrementing sequence.
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Microsoft SQL Server Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Microsoft SQL Server data type VARCHAR. By default, the server maps these data types as variable character (A*n*V).

The following table lists data type mappings based on the value of VARCHAR:

Microsoft SQL Server Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
VARCHAR (*n*)	*n* is an integer between 1 and 8000	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
VARBINARY (*n*)	*n* is an integer between 1 and 8000*m* = 2 * *n*	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE SQLMSS SET VARCHAR {ON|OFF}
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Microsoft SQL Server data type VARCHAR as variable-length alphanumeric (A*n*V). This is required for Unicode environments. ON is the default value.

`OFF`

Maps the Microsoft SQL Server data type VARCHAR as alphanumeric (A).

Enabling National Language Support

The SET parameter NCHAR indicates whether the character set is single-byte, double-byte, or triple-byte. The NCHAR setting affects the mapping of NCHAR and NVARCHAR data types.

The following chart lists data type mappings based on the value of NCHAR.

Microsoft SQL Server Data Type	Remarks	NCHAR SBCS		NCHAR DBCS		NCHAR TBCS	
NCHAR (*n*)	*n* is an integer between 1 and 4000 *d* = 2 * *n* *t* = 3 * *n*	A*n*	A*n*	A*d*	A*d*	A*t*	A*t*
NVARCHAR (*n*)	*n* is an integer between 1 and 4000 *d* = 2 * *n* *t* = 3 * *n*	A*n*	A*n*	A*d*	A*d*	A*t*	A*t*

Syntax: How to Enable National Language Support

The available parameters are:

```
ENGINE SQLMSS SET NCHAR {SBCS|DBCS|TBCS}
```

where:

SQLMSS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

SBCS

Indicates a single-byte character set. SBCS is the default value.

DBCS

Indicates a double-byte character set.

TBCS

Indicates a triple-byte character set.

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLMSS SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Support of Read-Only Fields

CREATE SYNONYM creates a field description with FIELDTYPE=R for Microsoft SQL Server columns created as TIMESTAMP or columns with the IDENTITY attribute. These fields are read-only. When executing a MAINTAIN or MODIFY procedure, the adapter suppresses all write operations against columns marked in the Master File with FIELDTYPE=R.

Example: Supporting a Read-Only Field

This example creates a table in which the first column has the IDENTITY property and the second column is a timestamp column:

```
CREATE TABLE TAB1
 (idproptab int IDENTITY (1,1), timstmp timestamp)
```

CREATE SYNONYM generates the following Master File for this table:

```
FILE=TAB1, SUFFIX=SQLMSS ,$
SEGNAME=TAB1, SEGTYPE=S0 ,$
FIELD=IDPROPTAB, idproptab, I11, I4,  MISSING=OFF, FIELDTYPE=R ,$
FIELD=TIMSTMP,   timstmp,   A16, A16, MISSING=ON,  FIELDTYPE=R ,$
```

Reporting Against a Microsoft SQL Server Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Microsoft SQL Server stored procedures and report against the procedure output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters: OUT parameters, and INOUT parameters in OUT mode, as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 1421.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 1425.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

A synonym describes the stored procedure parameters and answer set.

An answer set structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, *input parameters* refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception. If you know the internal logic of the procedure, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

Example: Synonym for Microsoft SQL Server Stored Procedure CustOrders

The following synonym describes a Microsoft SQL Server stored procedure with one input parameter, one output parameter, and one answer set containing four variables.

The Master File synonym is:

```
FILENAME=CUSTORDERS, SUFFIX=SQLMSS  , $
   SEGMENT=INPUT, SEGTYPE=S0, $
      FIELDNAME=@CUSTOMERID, ALIAS=P0001, USAGE=A5, ACTUAL=A5,
         MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
   SEGMENT=OUTPUT, SEGTYPE=S0, PARENT=INPUT, $
      FIELDNAME=@RETURN_VALUE, ALIAS=P0000, USAGE=I11, ACTUAL=I4, $
   SEGMENT=ANSWERSET1, SEGTYPE=S0, PARENT=INPUT, $
      FIELDNAME=ORDERID, ALIAS=OrderID, USAGE=I11, ACTUAL=I4, $
      FIELDNAME=ORDERDATE, ALIAS=OrderDate, USAGE=HYYMDs, ACTUAL=HYYMDs,
         MISSING=ON, $
      FIELDNAME=REQUIREDDATE, ALIAS=RequiredDate, USAGE=HYYMDs,
         ACTUAL=HYYMDs, MISSING=ON, $
      FIELDNAME=SHIPPEDDATE, ALIAS=ShippedDate, USAGE=HYYMDs,
         ACTUAL=HYYMDs, MISSING=ON, $
```

The Access File synonym is:

```
SEGNAME=INPUT, CONNECTION=ITarget, STPNAME=Northwind.dbo.CustOrders, $
SEGNAME=OUTPUT, STPRESORDER=0, $
SEGNAME=ANSWERSET1, STPRESORDER=1, $
```

Reference: Synonym Creation Parameters for Stored Procedures

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Select *Stored Procedures*.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check Use current database to use the database that has been set as the default database.
- Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

Before selecting a database, if Use current database is checked, uncheck it.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Note: For Db2, this applies to all platforms except IBM i.

Select

Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.

Name

The name of the synonym, which defaults to the stored procedure name.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters.

If all procedures have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Microsoft SQL Server Data Type Support* on page 1418.

Values

Select the check box for *every* parameter displayed for the specified procedure.

Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines:

- Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided.
- Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym creation purposes.

If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.

Creating a Report Against a Stored Procedure

You can report against a stored procedure answer set using the same facilities you use to report against a database table:

- **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 1426.
- **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 1425.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

`IF`

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters, or that there are no required parameters.

`WHERE`

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Microsoft SQL Server Environment

The Adapter for Microsoft SQL Server provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft SQL Server.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLMSS SET TIMEOUT {nn|0}
```

where:

SQLMSS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

0

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.

2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Specifying the Cursor Type

You can use the SET CURSORS command to specify the type of cursors for retrieval.

Syntax: How to Specify the Cursor Type

The available parameters are:

```
ENGINE SQLMSS SET CURSORS [CLIENT|SERVER]
```

where:

`CLIENT`

Uses Microsoft SQL Server client-side cursors for retrieving data. Client-side cursors normally demonstrate the best performance for data retrieval and benefit the Microsoft SQL Server process. However, except in TRANSACTIONS AUTOCOMMITTED mode, using client-side cursors prevents a server agent from simultaneously reading more than one answer set from the same instance of Microsoft SQL Server.

`SERVER`

Uses Microsoft SQL Server server-side cursors for retrieving data. Server-side cursors demonstrate lower performance than client cursors. However, setting a high FETCHSIZE factor (100 is the adapter default) improves performance dramatically making them almost as fast as client-side cursors. Client-side cursors are recommended wherever possible to take the load off the Microsoft SQL Server process.

`blank`

Uses client-side cursors in TRANSACTIONS AUTOCOMMITTED mode and server-side cursors otherwise. This value is the default.

Specifying the Login Wait Time

You can use the LOGINTIMEOUT command to specify the number of seconds the adapter will wait for a response from Microsoft SQL Server at connect time.

Note: For compatibility with previous releases of the adapter, TIMEOUT is available as a synonym for LOGINTIMEOUT.

Syntax: How to Specify the Login Wait Time

```
ENGINE SQLMSS SET LOGINTIMEOUT|TIMEOUT {nn|0}
```

where:

SQLMSS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is approximately 15 seconds.

0

Represents an infinite period to wait for login response.

Activating NONBLOCK Mode

The Adapter for Microsoft SQL Server has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax: How to Activate NONBLOCK Mode

The available parameters are:

```
ENGINE SQLMSS SET NONBLOCK {0|n}
```

where:

SQLMSS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.

- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

The available parameters are:

```
ENGINE SQLMSS SET PASSRECS {ON|OFF}
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Controlling Transactions

You can use the SET TRANSACTIONS command to controls how the adapter handles transactions.

Syntax: How to Control Transactions

The available parameters are:

```
ENGINE SQLMSS SET TRANSACTIONS {LOCAL|DISTRIBUTED|AUTOCOMMITTED}
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`LOCAL`

Indicates that the adapter implicitly starts a local transaction on each of the connections where any work is performed. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter commits or aborts the work on each connection consecutively. LOCAL is the default value.

`DISTRIBUTED`

Indicates that the adapter implicitly invokes Microsoft Distributed Transactions Coordinator (DTC) to create a single distributed transaction within which to perform all work on all the connections. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter invokes DTC to execute the two-phase commit or rollback protocol. For this purpose, the DTC service must be started on the machine where the server is running and also on all the machines where involved instances of Microsoft SQL Server reside.

This mode is recommended for read-write applications that perform updates on multiple connections simultaneously.

`AUTOCOMMITTED`

Indicates that each individual operation with Microsoft SQL Server is immediately committed (if successful) or rolled back (in case of errors) by the SQL Server. This is recommended for read-only applications for performance considerations. It is not recommended for read-write applications because in this mode it is impossible to roll back a logical unit of work that consists of several operations.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE SQLMSS SET ISOLATION {RU|RC|RR|SE|CH|CS}
```

where:

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

CH

Sets the transaction isolation level to Chaos.

CS

Sets the transaction isolation level to Cursor Stability, which is a synonym for Read Committed.

Microsoft SQL Server Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Optimizing Requests if a Virtual Field Contains Null Values

The SET OPTNOAGGR command provides finely-tuned control of adapter behavior for optimization. Users who for any reason wish to prevent passing aggregation to the RDBMS can use this command. An example of such a reason might be where NULL values occur in aggregated data with calculations. The SET OPTNOAGGR command causes the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is instead performed internally by the server while JOIN and SORT operations are handled by the RDBMS.

If any DEFINE field contains calculations with NULL fields then such operations cannot be translated to SQL and pass to DBMS because always return NULL. It has to be processed by FOCUS.

This can be achieved by SET OPTIMIZATION OFF.

However, in some cases it is preferable to use the off-load JOIN and SORT operation to DBMS for better performance while leaving AGGREGATION to FOCUS.

Syntax: How to Set Enhanced Aggregation Control

```
SQL SQLMSS SET OPT {AGGR|NOAGGR}
```

where:

AGGR

Directs the adapter to off-load aggregated DEFINE fields to the DBMS. This is the default setting.

NOAGGR

Directs the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is, instead, performed internally by the server, while JOIN and SORT operations are handled by the RDBMS. This setting can also be used to provide backwards compatibility for applications that were written based on the functionality of the previous release, when less SQL was off-loaded to the RDBMS. For example, when a calculation on aggregated fields may have contained NULL data that was not processed by the RDBMS NVL( ) function.

Example: Using IF-THEN_ELSE Optimization With a Condition That Is Always False

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.

- Self-referential expressions, such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

  Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.

- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

  Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting. Consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

The Adapter for Microsoft SQL Server supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE SQLMSS SET FETCHSIZE n
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE SQLMSS SET INSERTSIZE n
```

where:

`SQLMSS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE SQLMSS SET FASTLOAD [ON|OFF]
```

where:

SQLMSS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Uses the Bulk Insert API. ON is the default.

OFF

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Microsoft SQL Server.

For the Adapter for Microsoft SQL Server, the Bulk API is used automatically in LOADONLY mode. Measurements show that intermediate flushes do not affect performance; therefore, the behavior does not depend on the INSERTSIZE.

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Optimizing Non-Equality WHERE-Based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

`LEFT_OUTER`

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

`hostfile`

Is the host Master File.

`AT`

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

`hfld1`

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

`tag1`

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

`WITH` *`hfld2`*

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

`MULTIPLE`

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

`UNIQUE`

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations. It always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: **Conditions for WHERE-Based Outer Join Optimization**

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left Outer Join

The following request creates a left outer conditional join between two MSSQL data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE employee AT EMPLOYEEID
TO ALL FILE employeepayhistory AT EMPLOYEEID
     WHERE RATECHANGEDATE GT HIREDATE;
END
TABLE FILE employee
PRINT RATE
BY EMPLOYEEID
END
```

The WebFOCUS request is translated to a single MSSQL SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."EmployeeID", T1."HireDate", T2."EmployeeID",
T2."RateChangeDate", T2."Rate" FROM
AdventureWorks.HumanResources.Employee T1 LEFT OUTER JOIN
AdventureWorks.HumanResources.EmployeePayHistory T2 ON
(T2."RateChangeDate" > T1."HireDate") ) ORDER BY T1."EmployeeID";
```

Improving Optimizer Efficiency with Hints

DBMS Optimizer hints can be used to alter an execution plan. The adapter provides a setting which enable the TABLE command to place the hints at the end of the generated query for Microsoft SQL Server.

This occurs when the adapter constructs a single SELECT statement. It does not occur in the case of a FOCUS-managed Join when multiple SELECTs are generated.

To reverse the setting, use SET HINT without a hint_text parameter.

Syntax: How to Set Specific Hints

Use the following syntax to set specific hints:

```
SQL SQLMSS SET HINT OPTION (hint_text)
```

where

SQLMSS

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

```
OPTION (hint_text)
```

Is the text of the hint or hints combination. The end user is responsible for the syntax. Omitting OPTION (*hint_text*) resets the hint to none.

Example: **Setting an Microsoft SQL Hint**

```
SQL SQLMSS SET HINT OPTION (FAST 2)
TABLE FILE STXT31M
 PRINT *
 BY F01INT
 END
```

The WebFOCUS request is translated into a SELECT statement that incorporates the specified hint.

```
SELECT
   T1."F01INT",
   T1."F02CHAR_10",
   T1."F03VARCHAR_10"
FROM
   D999AIXPPC71XX_TTXT31M T1
ORDER BY
   T1."F01INT"
OPTION (FAST 2);
```

Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru

Microsoft SQL Server stored procedures are supported using SQL Passthru. These procedures need to be developed within Microsoft SQL Server using the CREATE PROCEDURE command.

The adapter supports stored procedures with input, output, and in-out parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets are returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to the rules of the underlying DBMS. Note that the examples shown in this section are SQL-based. See the DBMS documentation for rules, languages, and additional programming examples.

Syntax: How to Invoke a Stored Procedure

```
SQL SQLMSS EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

SQLMSS

Is the ENGINE suffix for Microsoft SQL Server.

procname

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

You can employ either SQL or SYS naming conventions to control the separator character used for interpreting multipart names, as described in Setting Naming Conventions.

parameter_specification

IN, OUT, and INOUT parameters are supported. Use the variation required by the stored procedure:

IN

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

OUT

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

INOUT

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

In this example, a user invokes a stored procedure, edaqa.test_proc01, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL SQLMSS EX edaqa.test_proc01 125,?,?/3.14,,'abc',,'xyz'
END
```

Example: Sample Stored Procedure

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (   OUT chSQLSTATE_OUT   CHAR(5),
                                  OUT intSQLCODE_OUT   INT,
                                  INOUT l_name char(20),
                                  INOUT f_name char(20))
    RESULT SETS 1
    LANGUAGE SQL
-------------------------------------------------------------------------
-- SQL Stored Procedure
-------------------------------------------------------------------------
P1: BEGIN
    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT
           EDAQA.NF29005.SSN5 AS SSN5,
           EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
           EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
           EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
           EDAQA.NF29005.SEX5 AS SEX5
        FROM
           EDAQA.NF29005
        WHERE
           (
             ( EDAQA.NF29005.LAST_NAME5 =  l_name )
        AND
             ( EDAQA.NF29005.FIRST_NAME5 = f_name )
           );
     -- Cursor left open for client application
    OPEN cursor1;
    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';
END P1   @
```

Reference: Capturing Application Errors in Stored Procedures

You can capture application errors using the RAISERROR method. Any application error that is issued by the stored procedure is available in the server variable &MSSMSGTXT.

Microsoft SQL Server Compatibility With ODBC

Release 5.2 and higher of the Adapter for Microsoft SQL Server is based on OLE DB, the Microsoft-recommended API for developing high-performance components. The adapter uses the OLE DB Provider for SQL Server (SQLOLEDB), a native, high-performance provider that directly accesses the SQL Server TDS protocol.

Releases of the adapter prior to 5.2 were based on the ODBC API. The current release of the adapter supports all the functionality of the earlier, ODBC-based versions. In addition, it introduces the following enhancements:

- Support for distributed transactions.
- Array Retrieval.
- Command execution timeout.
- Fast load. This feature, enabled automatically, improves the performance of load-only DataMigrator applications and MODIFY procedures.
- XA transaction support. For details, see *XA Support* on page 2689.

Differences between adapter functionality under OLE DB and ODBC exist in the following areas:

- Connection attributes.
- Cursors.
- Mapping of UNIQUEIDENTIFIER and BIT data types.

If you are using ODBC, be sure to review these topics.

Microsoft SQL Server Connection Attributes With ODBC

An ODBC data source declaration contains the following information: the instance of Microsoft SQL Server, the default database, the network libraries, and so on. In earlier, ODBC-based releases of the adapter, the SET CONNECTION_ATTRIBUTES command required only the name of an ODBC data source (User or System DNS) and authentication parameters.

Unlike ODBC data sources, OLE DB data sources directly reference instances of Microsoft SQL Server. To connect to an OLE DB data source, the adapter has to specify all the pertinent parameters at connection time. For this reason, the syntax of the SET CONNECTION_ATTRIBUTES command has been expanded to account for these parameters and also to provide the ability to declare more than one connection to the same instance of Microsoft SQL Server.

Microsoft SQL Server Cursors With ODBC

In earlier, ODBC-based releases of the adapter, the SET parameter CONCUR is used to control the ODBC cursor type. The CONCUR parameter has two settings: RONLY results in client-side cursors, and ROWVER forces cursors to be server-side. Release 5.2 of the adapter takes advantage of OLE DB's ability to directly control the cursor type using a CURSOR parameter. The SET parameter CURSOR has two settings: CLIENT and SERVER. By default, the adapter assigns cursor properties that provide maximum performance for the given environment.

Mapping of UNIQUEIDENTIFIER and BIT Data Types

OLE DB maps the following data types differently than ODBC:

- UNIQUEIDENTIFER is mapped to 38 characters. ODBC maps this data type to 36 characters. This difference results from the fact that OLE DB encloses the value in braces ({ }).
- BIT is mapped to -1 for TRUE and 0 for FALSE. ODBC maps TRUE to 1.

Chapter 60

Using the Adapter for Microsoft SQL Server Analysis Services (SSAS)

The Adapter for SQL Server Analysis Services (SSAS) allows applications to access SQL Server Analysis Services data sources. The Adapter for TM1 is a special configuration of the Adapter for SQL Server Analysis Services. For information about this configuration, see *Configuring the Adapter for TM1* on page 1450.

The adapter converts application requests into Multidimensional Expressions (MDX), which it submits to SQL Server Analysis Services and returns the resulting data to the WebFOCUS Reporting Server.

In this chapter:

Preparing the SQL Server Analysis Services (SSAS) Environment

The Adapter for SQL Server Analysis Services minimally requires the installation of the Microsoft OLE DB Provider for Analysis Services version 10.0 and up, which is part of the Microsoft Data Access Components (MDAC) installation.

Procedure: How to Set Up the Environment

Microsoft OLE DB Provider for Analysis Services 9.0, 10.0, or 11.0 must be installed in your system.

The Adapter for SQL Server Analysis Services provides read-only access to analytical data stored in cubes. The adapter is an OLE DB for OLAP consumer. It employs Multidimensional Expressions (MDX) language to access aggregated or rolled-up data used for decision-support processing.

You can use any server front-end technology with the WebFOCUS reporting engine to access the OLAP data retrieved by the server for Windows. This makes your OLAP data available to your entire enterprise. Additionally, data from SSAS cubes can be joined with data from any other supported data source, providing additional information to your analytical process.

Setting SQL Server Analysis Services (SSAS) Security

There are three methods by which you can be authenticated when connecting to SQL Server Analysis Services:

Operating System (Trusted) mode. The user ID and password used to log on to the operating system are automatically used to connect to the Analytical Engine. The server data access agent impersonates an operating system user according to the server deployment mode. The agent process establishes a connection to a SSAS based on the impersonated operating system user credentials.

SSAS recognizes the security restrictions that the OLAP Database Administrator specifies and applies them to catalogs and cubes. This includes the role of security on the database and the application level. Although the Server for Windows establishes the connection to SSAS, an impersonation of the client is used to maintain access privileges.

- **Security off.** If the Server is running with security off, the user ID and password of user that started the Reporting Server is passed to the Analytical Engine in order to establish a connection and access rights. This is the logon ID specified for the Windows or Windows Workstation after it is first started.
- **Security on.** If the Server is running with security on, the system credentials of the client connecting to the server is passed to the Analytical Engine in order to establish a connection and access privileges. This process is called impersonation.

Explicit mode. The user ID and password are explicitly specified for each connection and passed to SQL Server Analysis Services, at connection time, for authentication

Password passthru mode. The user ID and password are explicitly specified for each connection and passed to SQL Server Analysis Services, at connection time, for authentication.

Accessing SQL Server Analysis Services

Using the standard rules for deploying OLE DB, the server supports connections to:

- Local SQL Server Analysis Services.

- Remote SQL Server Analysis Services. (To connect to a remote SSAS instance, you must have Microsoft OLE DB Provider for Analysis Services 9.0 installed in your system.)

Configuring the Adapter for SQL Server Analysis Services

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one SQL Server Analysis Services.

You can declare connections to more than one Analytical Engine by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued. You can include those commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands, the first command sets the default SQL Server Analysis Services to be used.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for SQL Server Analysis Services (SSAS)

The SQL Server Analysis Services adapter is under the *OLAP* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server

Name of the machine where SQL Server Analysis Services (SSAS) is running. If that machine has more than one instance of SQL Server Analysis Services (SSAS) installed, provide the server name and the instance name as follows: server\instance.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Default Database

Must be specified only when accessing Cognos TM1. For example, SData.

Additional connection string keywords (Optional)

Are extended connection parameters (*provider string*) for specification of such parameters as EffectiveUserName for SSAS or CAMNamespace for TM1. These values should be entered as keyword=value pairs separated by semicolons (;). For example, EffectiveUserName=Domain\UserID (SSAS) or CAMNamespace=NTLM_NAMESPACE (TM1)

Syntax:

How to Declare Connection Attributes Manually

Trusted authentication. The adapter connects to SQL Server Analysis Services (SSAS) as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE SSAS SET CONNECTION_ATTRIBUTES [connection]server/
```

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to SQL Server Analysis Services (SSAS), at connection time, for authentication as a standard login.

```
ENGINE SSAS SET CONNECTION_ATTRIBUTES [connection]server/user,pwd
```

Password passthru authentication. The user ID and password received from the client application are passed to SQL Server Analysis Services (SSAS), at connection time, for authentication as a standard login.

```
ENGINE SSAS SET CONNECTION_ATTRIBUTES [connection]server
```

where:

SSAS

Indicates the adapter. You can omit this value if you previously issued the SET SQL ENGINE command.

connection

Is a logical name used to identify this particular set of attributes.

If you plan to have only one connection to SQL Server Analysis Services, Is the server name used as a connect descriptor to the SQL Server Analysis ServicesServer across the network..

server

Is the machine where SQL Server Analysis Services is running.

user

Is the primary authorization ID by which you are known to SQL Server Analysis Services.

pwd

Is the password associated with the primary authorization ID.

Configuring the Adapter for TM1

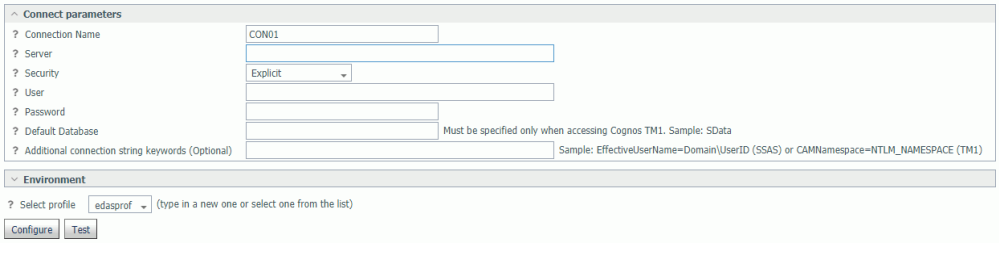
The Adapter for TM1 is a special configuration of the Adapter for SQL Server Analysis Services. In order to access TM1, theTM1 client software (which includes the TM1OLAP provider) must be installed.

Configure the Adapter for TM1 by selecting *TM1OLAP Provider* on the *Change Settings for SQL Server Analysis Services* page of the Web Console. You must also provide the default database and, possibly, additional connection string parameters on the *Change Connect Parameters for SQL Server Analysis Services* Web Console page. In addition to Standard Authentication (login) security, when the Cognos TM1 server requests the user name and password, and validates the login information against the login information stored in the internal security cube, the Adapter for TM1 supports Cognos Access Manager (CAM) authentication and Integrated Login.

CAM Authentication Support

To configure Cognos Access Manager (CAM) authenticaion, you must add the CAMNamespace parameter to the connection string.

You can configure this parameter on the Web Console. To configure the Adapter for TM1, use the Add Connection page for the Adapter for SQL Server Analysis Services, as shown in the following image.



For TM1, the catalog to be accessed must be part of the connection string. Enter the catalog name in the *Default Database* text box in order to add it to the connection string.

Enter the CAMNamespace parameter in the *Additional connection string keywords* text box. It will be added to the connection string as the ProviderString parameter when you press *Configure*. For example, if you enter CAMNamespace=NTLM_NAMESPACE, a connection string similar to the following is generated in the Server Profile (edasprof.prf):

```
ENGINE SSAS SET CONNECTION_ATTRIBUTES CON1
SQL2012x64-02/user1,pwd1;SData:CAMNamespace=NTLM_NAMESPACE
```

The ENGINE SSAS SET PROVIDER TM1OLAP command must be in effect when using the Adapter for TM1. This command can be issued in the Server Profile (edasprof.prf).

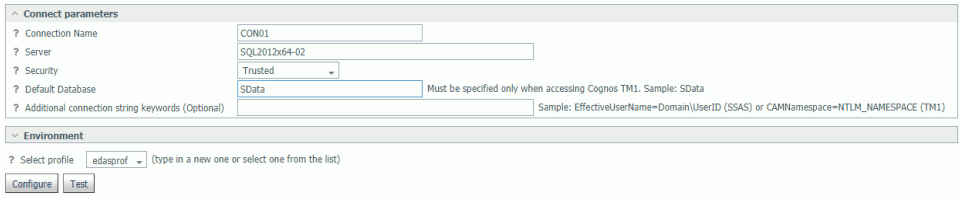
Integrated Login Support

Integrated Login enables you to use Microsoft Windows network authentication to control access to IBM Cognos TM1 data.

In this security model, user and group Microsoft Windows login information has to be moved into the Cognos TM1 database. Integrated Login matches the domain-qualified name you use to sign in to Microsoft Windows with a name stored in the internal database.

Integrated Login is supported on Microsoft Windows only.

You can configure Integrated Login in the Web Console, using the Add Connection page of the Adapter for SQL Server Analysis Services. When you configure the Adapter for TM1, select a trusted connection, as shown in the following image.



The connection string must specify the name of the TM1 catalog to be accessed. Enter the catalog name in the *Default Database* text box. A connection string similar to the following will be generated in the Server Profile (edasprof.prf).

```
ENGINE SSAS SET CONNECTION_ATTRIBUTES CON1
SQL2012x64-02/,;SData
```

Since the connection is configured as trusted, no user ID or password is included in the connection string. The Integrated SSPI login causes the credentials from the Windows login to be used to access TM1.

The ENGINE SSAS SET PROVIDER TM1OLAP command must be in effect when using the Adapter for TM1. This command can be issued in the Server Profile (edasprof.prf).

Managing SQL Server Analysis Services Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the SQL Server Analysis Services data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Analytical Engine cube that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

For details on how to create a synonym through the Web Console, see the *Server Configuration and Operation* manual for your platform.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for SQL Server Analysis Services (SSAS)

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select catalog

You can select a catalog from the drop-down list. The default selection is *All Catalogs*.

Filter by Cube Name

If you wish to filter your selections, click this check box and enter a cube name in the input box.

You can enter a full cube name or use a wildcard character (%) as needed, at the beginning and/or end of the string. For example, enter: ABC% to select cubes that begin with the letters ABC; %ABC to select cubes that end with the letters ABC; %ABC% to select cubes that contain the letters ABC at the beginning, middle, or end.

Include Attribute (System-Enabled) Hierarchies

To include all hierarchies, not just user-defined hierarchies, in the synonym, select the *Include Attribute (System-Enabled) Hierarchies* check box.

Generate Qualified Field Names

To generate qualified field names (field names qualified with the Master File and segment names) in the synonym, click the *Generate Qualified Field Names* check box.

Create Fields for Member Unique Names

To create fields in the synonym for member unique names, select the *Create Fields for Member Unique Names* check box.

Present Parent/child Hierarchies as

Select how you want parent/child hierarchies represented in the synonym:

- As parent/child hierarchies, the default. This option creates a set of fields for each parent/child hierarchy that describe a member's position in the hierarchy. To report from a parent/child hierarchy use the BY HIERARCHY syntax to automatically retrieve, display, and format portions of the hierarchy with appropriate indentations to identify the hierarchical relationships between the members.
- As level hierarchies. This option creates separate fields for each hierarchy level. To report from this type of hierarchy, specify the field name for each level of the hierarchy you want to display as a BY field in the request.

Field Name Case

Select whether you want mixed-case or uppercase field names. Mixed-case is the default value.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Select cubes

Select the cubes for which you wish to create synonyms:

- To select all cubes in the list, select the check box to the left of the *Default Synonym Name* column heading.
- To select specific cubes, select the corresponding check boxes.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Example: Creating a Synonym

To generate a synonym for the Sales cube in the FoodMart 2000 Data Warehouse, enter the following information on the Create Synonym panes of the Web Console or the Data Management Console:

1. On the first Create Synonym pane, select *FoodMart 2000* from the *Select catalog* drop-down list.
2. Click the *Filter by Cube Name* check box.
3. Enter *S%* in the Cube Name field.
4. Click the *Next* button.
5. On the second Create Synonym pane, select *Sales* from the list of displayed cubes in the *FoodMart 2000* catalog.
6. Click *Create Synonym*. The synonym is created and added under the specified application directory (The default is ibisamp).

   A status window displays that the synonym was created successfully.
7. Open the *ibisamp* application folder in the navigation pane and right-click the synonym *Sales*.
8. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File

```
FILENAME=Sales, SUFFIX=SSAS     , $
  SEGMENT=SALES, SEGTYPE=S0, $
    FIELDNAME=Unit_Sales, ALIAS='Unit Sales', USAGE=D18.2C, ACTUAL=D8,
      MISSING=ON,
      TITLE='Unit Sales', MEASURE_GROUP=Sales,
      PROPERTY=MEASURE,  $
```

```
    FIELDNAME=Store_Cost, ALIAS='Store Cost', USAGE=D18.2C, ACTUAL=D8,
      MISSING=ON,
      TITLE='Store Cost', MEASURE_GROUP=Sales,
      PROPERTY=MEASURE,  $
    FIELDNAME=Store_Sales, ALIAS='Store Sales', USAGE=P20.2MB, ACTUAL=P16,
      MISSING=ON,
      TITLE='Store Sales', MEASURE_GROUP=Sales,
      PROPERTY=MEASURE,  $
    FIELDNAME=Sales_Count, ALIAS='Sales Count', USAGE=I11, ACTUAL=I4,
      MISSING=ON,
      TITLE='Sales Count', MEASURE_GROUP=Sales,
      PROPERTY=MEASURE,  $
    FIELDNAME=Store_Sales_Net, ALIAS='Store Sales Net', USAGE=D18.2C,
ACTUAL=D8,
      MISSING=ON,
      TITLE='Store Sales Net', MEASURE_GROUP=Sales,
      PROPERTY=MEASURE,  $
    FIELDNAME=Profit, ALIAS=Profit, USAGE=D18.2C, ACTUAL=D8,
      MISSING=ON,
      TITLE='Profit', MEASURE_GROUP=Undefined,
      PROPERTY=MEASURE,  $
    FIELDNAME=Sales_Average, ALIAS='Sales Average', USAGE=D18.2C, ACTUAL=D8,
      MISSING=ON,
      TITLE='Sales Average', MEASURE_GROUP=Undefined,
      PROPERTY=MEASURE,  $
 DIMENSION=[Customers], CAPTION='Customers', $
  HIERARCHY=[Customers], CAPTION='Customers', HRY_DIMENSION=[Customers], $
    FIELDNAME=Country, ALIAS=Country, USAGE=A6, ACTUAL=A6,
      TITLE='Country',
      WITHIN='*[Customers]',
      PROPERTY=CAPTION,  $
```

```
    FIELDNAME=State_Province, ALIAS='State Province', USAGE=A9, ACTUAL=A9,
      TITLE='State Province',
      WITHIN=Country,
      PROPERTY=CAPTION,  $
    FIELDNAME=City, ALIAS=City, USAGE=A14, ACTUAL=A14,
      TITLE='City',
      WITHIN=State_Province,
      PROPERTY=CAPTION,  $
    FIELDNAME=Name, ALIAS=Name, USAGE=A26, ACTUAL=A26,
      TITLE='Name',
      WITHIN=City,
      PROPERTY=CAPTION,  $
$ LEVEL [Customers].[Name] PROPERTIES
    FIELDNAME=Date_Accnt_Opened, ALIAS='Date Accnt Opened', USAGE=A10,
ACTUAL=A10,
      TITLE='Date Accnt Opened',
      REFERENCE=Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Education, ALIAS=Education, USAGE=A19, ACTUAL=A19,
      TITLE='Education',
      REFERENCE=Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Gender, ALIAS=Gender, USAGE=A1, ACTUAL=A1,
      TITLE='Gender',
      REFERENCE=Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Marital_Status, ALIAS='Marital Status', USAGE=A1, ACTUAL=A1,
      TITLE='Marital Status',
      REFERENCE=Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Member_Card, ALIAS='Member Card', USAGE=A6, ACTUAL=A6,
      TITLE='Member Card',
      REFERENCE=Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Yearly_Income, ALIAS='Yearly Income', USAGE=A13, ACTUAL=A13,
      TITLE='Yearly Income',
      REFERENCE=Name, PROPERTY=ATTRIBUTE,  $
 DIMENSION=[Education Level], CAPTION='Education Level', $
  HIERARCHY=[Education Level], CAPTION='Education Level',
HRY_DIMENSION=[Education Level], $
    FIELDNAME=Education_Level1, ALIAS='Education Level', USAGE=A19,
ACTUAL=A19,
      TITLE='Education Level',
      WITHIN='*[Education Level]',
      PROPERTY=CAPTION,  $
 DIMENSION=[Gender], CAPTION='Gender', $
  HIERARCHY=[Gender], CAPTION='Gender', HRY_DIMENSION=[Gender], $
    FIELDNAME=Gender11, ALIAS=Gender, USAGE=A1, ACTUAL=A1,
      TITLE='Gender',
      WITHIN='*[Gender]',
      PROPERTY=CAPTION,  $
```

```
DIMENSION=[Marital Status], CAPTION='Marital Status', $
  HIERARCHY=[Marital Status], CAPTION='Marital Status',
HRY_DIMENSION=[Marital Status], $
    FIELDNAME=Marital_Status11, ALIAS='Marital Status', USAGE=A1, ACTUAL=A1,
      TITLE='Marital Status',
      WITHIN='*[Marital Status]',
      PROPERTY=CAPTION,  $
 DIMENSION=[Product], CAPTION='Product', $
  HIERARCHY=[Product], CAPTION='Product', HRY_DIMENSION=[Product], $
    FIELDNAME=Product_Family, ALIAS='Product Family', USAGE=A14, ACTUAL=A14,
      TITLE='Product Family',
      WITHIN='*[Product]',
      PROPERTY=CAPTION,  $
    FIELDNAME=Product_Department, ALIAS='Product Department', USAGE=A19,
ACTUAL=A19,
      TITLE='Product Department',
      WITHIN=Product_Family,
      PROPERTY=CAPTION,  $
    FIELDNAME=Product_Category, ALIAS='Product Category', USAGE=A20,
ACTUAL=A20,
      TITLE='Product Category',
      WITHIN=Product_Department,
      PROPERTY=CAPTION,  $
    FIELDNAME=Product_Subcategory, ALIAS='Product Subcategory', USAGE=A18,
ACTUAL=A18,
      TITLE='Product Subcategory',
      WITHIN=Product_Category,
      PROPERTY=CAPTION,  $
    FIELDNAME=Brand_Name, ALIAS='Brand Name', USAGE=A13, ACTUAL=A13,
      TITLE='Brand Name',
      WITHIN=Product_Subcategory,
      PROPERTY=CAPTION,  $
    FIELDNAME=Product_Name, ALIAS='Product Name', USAGE=A43, ACTUAL=A43,
      TITLE='Product Name',
      WITHIN=Brand_Name,
      PROPERTY=CAPTION,  $
```

```
$ LEVEL [Product].[Product Name] PROPERTIES
    FIELDNAME=Cases_Per_Pallet, ALIAS='Cases Per Pallet', USAGE=I2,
ACTUAL=I4,
      TITLE='Cases Per Pallet',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Gross_Weight, ALIAS='Gross Weight', USAGE=D6.2, ACTUAL=D8,
      TITLE='Gross Weight',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Low_Fat, ALIAS='Low Fat', USAGE=I2, ACTUAL=I4,
      TITLE='Low Fat',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Net_Weight, ALIAS='Net Weight', USAGE=D6.2, ACTUAL=D8,
      TITLE='Net Weight',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Recyclable_Package, ALIAS='Recyclable Package', USAGE=I2,
ACTUAL=I4,
      TITLE='Recyclable Package',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Shelf_Depth, ALIAS='Shelf Depth', USAGE=D6.2, ACTUAL=D8,
      TITLE='Shelf Depth',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Shelf_Height, ALIAS='Shelf Height', USAGE=D6.2, ACTUAL=D8,
      TITLE='Shelf Height',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Shelf_Width, ALIAS='Shelf Width', USAGE=D6.2, ACTUAL=D8,
      TITLE='Shelf Width',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Sku, ALIAS=Sku, USAGE=D12, ACTUAL=D8,
      TITLE='Sku',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Srp, ALIAS=Srp, USAGE=D5.2, ACTUAL=D8,
      TITLE='Srp',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Units_Per_Case, ALIAS='Units Per Case', USAGE=I2, ACTUAL=I4,
      TITLE='Units Per Case',
      REFERENCE=Product_Name, PROPERTY=ATTRIBUTE,  $
 DIMENSION=[Promotion Media], CAPTION='Promotion Media', $
  HIERARCHY=[Promotion Media], CAPTION='Promotion Media',
HRY_DIMENSION=[Promotion Media], $
    FIELDNAME=Media_Type, ALIAS='Media Type', USAGE=A23, ACTUAL=A23,
      TITLE='Media Type',
      WITHIN='*[Promotion Media]',
      PROPERTY=CAPTION,  $
```

```
DIMENSION=[Promotions], CAPTION='Promotions', $
  HIERARCHY=[Promotions], CAPTION='Promotions', HRY_DIMENSION=[Promotions],
$
    FIELDNAME=Promotion_Name, ALIAS='Promotion Name', USAGE=A23, ACTUAL=A23,
      TITLE='Promotion Name',
      WITHIN='*[Promotions]',
      PROPERTY=CAPTION,  $
 DIMENSION=[Store], CAPTION='Store', $
  HIERARCHY=[Store], CAPTION='Store', HRY_DIMENSION=[Store], $
  HIERARCHY=[Store Size in SQFT], CAPTION='Store Size in SQFT',
HRY_DIMENSION=[Store], $
  HIERARCHY=[Store Type], CAPTION='Store Type', HRY_DIMENSION=[Store], $
    FIELDNAME=Store_Country, ALIAS='Store Country', USAGE=A6, ACTUAL=A6,
      TITLE='Store Country',
      WITHIN='*[Store]',
      PROPERTY=CAPTION,  $
    FIELDNAME=Store_State, ALIAS='Store State', USAGE=A9, ACTUAL=A9,
      TITLE='Store State',
      WITHIN=Store_Country,
      PROPERTY=CAPTION,  $
    FIELDNAME=Store_City, ALIAS='Store City', USAGE=A13, ACTUAL=A13,
      TITLE='Store City',
      WITHIN=Store_State,
      PROPERTY=CAPTION,  $
    FIELDNAME=Store_Name, ALIAS='Store Name', USAGE=A8, ACTUAL=A8,
      TITLE='Store Name',
      WITHIN=Store_City,
      PROPERTY=CAPTION,  $
$ LEVEL [Store].[Store Name] PROPERTIES
    FIELDNAME=Store_Manager, ALIAS='Store Manager', USAGE=A8, ACTUAL=A8,
      TITLE='Store Manager',
      REFERENCE=Store_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Store_Sqft, ALIAS='Store Sqft', USAGE=A5, ACTUAL=A5,
      TITLE='Store Sqft',
      REFERENCE=Store_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Store_Type, ALIAS='Store Type', USAGE=A19, ACTUAL=A19,
      TITLE='Store Type',
      REFERENCE=Store_Name, PROPERTY=ATTRIBUTE,  $
    FIELDNAME=Store_Sqft1, ALIAS='Store Sqft', USAGE=A5, ACTUAL=A5,
      TITLE='Store Sqft',
      WITHIN='*[Store Size in SQFT]',
      PROPERTY=CAPTION,  $
    FIELDNAME=Store_Type11, ALIAS='Store Type', USAGE=A19, ACTUAL=A19,
      TITLE='Store Type',
      WITHIN='*[Store Type]',
      PROPERTY=CAPTION,  $
```

```
DIMENSION=[Time], CAPTION='Time', $
  HIERARCHY=[Time], CAPTION='Time', HRY_DIMENSION=[Time], $
    FIELDNAME=Year, ALIAS=Year, USAGE=I4, ACTUAL=I4,
      TITLE='Year',
      WITHIN='*[Time]',
      PROPERTY=CAPTION,  $
    FIELDNAME=Quarter, ALIAS=Quarter, USAGE=A2, ACTUAL=A2,
      TITLE='Quarter',
      WITHIN=Year,
      PROPERTY=CAPTION,  $
    FIELDNAME=Month, ALIAS=Month, USAGE=I2, ACTUAL=I4,
      TITLE='Month',
      WITHIN=Quarter,
      PROPERTY=CAPTION,  $
 DIMENSION=[Yearly Income], CAPTION='Yearly Income', $
  HIERARCHY=[Yearly Income], CAPTION='Yearly Income', HRY_DIMENSION=[Yearly
Income], $
    FIELDNAME=Yearly_Income11, ALIAS='Yearly Income', USAGE=A13, ACTUAL=A13,
      TITLE='Yearly Income',
      WITHIN='*[Yearly Income]',
      PROPERTY=CAPTION,  $
 MEASUREGROUP=Sales, $
   MEASGRPDIM=[Customers], $
   MEASGRPDIM=[Education Level], $
   MEASGRPDIM=[Gender], $
   MEASGRPDIM=[Marital Status], $
   MEASGRPDIM=[Product], $
   MEASGRPDIM=[Promotion Media], $
   MEASGRPDIM=[Promotions], $
   MEASGRPDIM=[Store], $
   MEASGRPDIM=[Time], $
   MEASGRPDIM=[Yearly Income], $
 MEASUREGROUP=Undefined, CAPTION='(Undefined)', $
```

Note: Master Files should not be altered. Manipulating the Master File causes incorrect query results.

The Measures dimension can contain calculated and non-calculated members. Calculated members in the Measures dimension will contain a Reference defining the calculated member and a Property. Some properties are mandatory while others may be user-defined.

Generated Access File

```
SEGNAME=SALES,
  CUBENAME=Sales,
  SCHEMA=,
  CATALOG=FoodMart,
  DATASOURCE=CON1, $
```

Reference: Selecting Valid Dimensions for a Measure

A data source view may have multiple fact tables as well as multiple dimension tables. Each fact table contains records for a specific set of measures called a *measure group*. A particular measure group may not have a relationship to every dimension table, and if you select a measure/dimension combination where there is no relationship, the returned data might be meaningless.

Starting in Version 7 Release 7.04, when you create a synonym, each measure group and its related dimensions are listed at the bottom of the synonym. In addition, the FIELD declaration each measure in the Master File has a MEASURE_GROUP attribute identifying the measure group to which it belongs. Using this information, the WebFOCUS tools will gray out the dimensions that are not valid for each measure group so that you can make a valid measure/dimension selection for your report.

For example, the following portion of the Adventure_Works Master File shows the Internet_Sales FIELD declaration with its MEASURE_GROUP attribute:

```
FIELDNAME=Internet_Sales_Amount, ALIAS='Internet Sales Amount',
USAGE=P20.2MB, ACTUAL=P16,
      MISSING=ON,
      TITLE='Internet Sales Amount', MEASURE_GROUP=Internet Sales,
      PROPERTY=MEASURE,  $
```

The following portion of the Adventure_Works Master File shows the Internet_Sales measure group along with the dimensions that are valid for this measure group:

```
  MEASUREGROUP=Internet Sales, $
   MEASGRPDIM=[Customer], $
   MEASGRPDIM=[Date], $
   MEASGRPDIM=[Delivery Date], $
   MEASGRPDIM=[Internet Sales Order Details], $
   MEASGRPDIM=[Product], $
   MEASGRPDIM=[Promotion], $
   MEASGRPDIM=[Sales Reason], $
   MEASGRPDIM=[Sales Territory], $
   MEASGRPDIM=[Ship Date], $
```

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the following options.

Option	Description
Open	Opens the Master File for viewing and editing using a graphical interface. If an Access file is used it will be also available.
Test Query	Opens the Data Assist tool with the specified synonym selected.
Edit as Text	Enables you to view and manually edit the Master File. **Note:** To update the synonym, it is strongly recommended that you use the graphical interface provided by the *Open* option, rather than manually editing the Master File.
Edit Access File as Text	Enables you to view and manually edit the Access File. **Note:** This option is available only when an Access File is created as part of the synonym.
Sample Data	Retrieves up to 50 rows from the associated data source.
Quick Copy	Opens the Quick Copy page to select the properties for the copy.
Custom Copy	Opens the Data Assist tool with the specified synonym selected.
Impact Analysis	Generates a report showing where this synonym is stored and used, with links to the synonym instances. Impact Analysis reports enable you to evaluate changes before they are made by showing which components will be affected. See the *Server Administration* manual for details about Impact Analysis reports.
Dependencies Analysis	Generates a report showing information about the synonym and other synonyms and objects that are referenced within it.
Download	Enables you to open the Master File or Access File in a text editor or save it.
Copy	Copies the synonym to the clipboard.

Option	Description
Delete	Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.
Cut	Deletes the synonym and places it on the clipboard.
Rename	Allows you to rename the synonym.
Properties	Displays the properties of the synonym, including physical location, last modified date, description, and privileges.

Reference: Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
CUBENAME	Name of the OLAP cube.
SCHEMA	Schema that describes the cube.
DATASOURCE	SQL Server Analysis Services name.
CATALOG	Name of the data source (catalog) the OLAP cube is using.

Converting Alphanumeric Dates to WebFOCUS Dates

Microsoft Analysis Server natively stores data (such as, member names and captions) in time dimensions in alphanumeric format without applying any particular standard, and it allows you to create members with names consisting of any combination of components such as year, quarter, and month, with any delimiter and in many languages. In a sorted report, if such data is sorted alphabetically, the sequence does not make business sense. To ensure adequate sorting, aggregation, and reporting on date fields, WebFOCUS converts the alphanumeric dates into standard WebFOCUS date format using a conversion pattern that you can specify in the Master File attribute called DATEPATTERN.

Each element in the pattern is either a constant character which must appear in the actual input or a variable that represents a date component. You must edit the USAGE attribute in the Master File so that it accounts for the date elements in the date pattern. The maximum length of the DATEPATTERN string is 64 characters.

Note: You can use the Synonym Editor to edit a date pattern. This is recommended in order to avoid inadvertant syntax errors in the pattern.

Specifying Variables in a Date Pattern

The valid date components (variables) are year, quarter, month, day, and day of week. In the date pattern, variables are enclosed in square brackets (these brackets are not part of the input or output. Note that if the data contains brackets, you must use an escape character in the date pattern to distinguish the brackets in the data from the brackets used for enclosing variables).

Syntax: **How to Specify Years in a Date Pattern**

`[YYYY]`

Specifies a 4-digit year.

`[YY]`

Specifies a 2-digit year.

`[yy]`

Specifies a zero-suppressed 2-digit year (For example, 8 for 2008).

`[by]`

Specifies a blank-padded 2-digit year.

Reference: **Specify Month Numbers in a Date Pattern**

`[MM]`

Specifies a 2-digit month number.

`[mm]`

Specifies a zero-suppressed month number.

`[bm]`

Specifies a blank-padded month number.

Syntax: How to Specify Month Names in a Date Pattern

`[MON]`

Specifies a 3-character month name in uppercase.

`[mon]`

Specifies a 3-character month name in lowercase.

`[Mon]`

Specifies a 3-character month name in mixed-case.

`[MONTH]`

Specifies a full month name in uppercase.

`[month]`

Specifies a full month name in lowercase.

`[Month]`

Specifies a full month name in mixed-case.

Syntax: How to Specify Days of the Month in a Date Pattern

`[DD]`

Specifies a 2-digit day of the month.

`[dd]`

Specifies a zero-suppressed day of the month.

`[bd]`

Specifies a blank-padded day of the month.

Syntax: How to Specify Julian Days in a Date Pattern

`[DDD]`

Specifies a 3-digit day of the year.

`[ddd]`

Specifies a zero-suppressed day of the year.

`[bdd]`

Specifies a blank-padded day of the year.

Syntax: **How to Specify Day of the Week in a Date Pattern**

`[WD]`

Specifies a 1-digit day of the week.

`[DAY]`

Specifies a 3-character day name, uppercase.

`[day]`

Specifies a 3-character day name, lowercase.

`[Day]`

Specifies a 3-character day name, mixed-case.

`[WDAY]`

Specifies a full day name, uppercase.

`[wday]`

Specifies a full day name, lowercase.

`[Wday]`

Specifies a full day name, mixed-case.

For the day of the week, the WEEKFIRST setting defines which day is day 1.

Syntax: **How to Specify Quarters in a Date Pattern**

`[Q]`

Specifies a 1-digit quarter number (1, 2, 3, or 4).

For a string like Q2 or Q02, use constants before [Q], for example, Q0[Q].

Specifying Constants in a Date Pattern

Between the variables, you can insert any constant values.

If you want to insert a character that would normally be interpreted as part of a variable, use the backslash character as an escape character. For example:

- Use \[to specify a left square bracket constant character.
- Use \\ to specify a backslash constant character.

For a single quotation mark, use two consecutive single quotation marks ('').

Example: Sample Date Patterns

If the date in the data source is of the form CY 2001 Q1, the DATEPATTERN attribute is:

```
DATEPATTERN = 'CY [YYYY] Q[Q]'
```

If the date in the data source is of the form Jan 31, 01, the DATEPATTERN attribute is:

```
DATEPATTERN = '[Mon] [DD], [YY]'
```

If the date in the data source is of the form APR-06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON]-[YY]'
```

If the date in the data source is of the form APR - 06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] - [YY]'
```

If the date in the data source is of the form APR '06, the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] ''[YY]'
```

If the date in the data source is of the form APR [06], the DATEPATTERN attribute is:

```
DATEPATTERN = '[MON] \[[YY]\]' (or '[MON] \[[YY]]'
```

Note the right square bracket does not have to be escaped.

Example: Sorting By an Alphanumeric Date

The following request against the Adventure Works cube sorts by the Month field:

```
TABLE FILE adventure_works_mixed_case
SUM
Internet_Sales_Amount
BY Month
WHERE Calendar_Year EQ 'CY 2001' OR 'CY 2004'
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET HTMLCSS ON
END
```

The default synonym has the following declaration for the Month field:

```
 FIELDNAME=Month, ALIAS=Month, USAGE=A14, ACTUAL=A14,
      TITLE='Month',
      WITHIN='Calendar_Quarter',
      PROPERTY=CAPTION,  $
```

The output shows that the months also have a year component and are sorted in alphabetical order, not month order:

Month	Internet Sales Amount
April 2004	$1,608,750.53
August 2001	$506,191.69
December 2001	$755,527.89
February 2004	$1,462,479.83
January 2004	$1,340,244.95
July 2001	$473,388.16
July 2004	$50,840.63
June 2004	$1,949,361.11
March 2004	$1,480,905.18
May 2004	$1,878,317.51
November 2001	$543,993.40
October 2001	$513,329.47
September 2001	$473,943.03

Now, make the following changes to the declaration for the Month field:

- Change the USAGE to a WebFOCUS date format with the same date components as the alphanumeric date (MtrYY).
- Add a DATEPATTERN attribute that describes the alphanumeric date stored in the field.
  - Variable: mixed case full month name, [Month].
  - Literal: blank space.
  - Variable: four-digit year, [YYYY].

```
FIELDNAME=Month, ALIAS=Month, USAGE=MtrYY, ACTUAL=A14,
      TITLE='Month',
      WITHIN='Calendar_Quarter',
      PROPERTY=CAPTION,
      DATEPATTERN = '[Month] [YYYY]',$
```

Re-running the request with these changes produces the following output in which the months are sorted correctly:

Month	Internet Sales Amount
July, 2001	$473,388.16
August, 2001	$506,191.69
September, 2001	$473,943.03
October, 2001	$513,329.47
November, 2001	$543,993.40
December, 2001	$755,527.89
January, 2004	$1,340,244.95
February, 2004	$1,462,479.83
March, 2004	$1,480,905.18
April, 2004	$1,608,750.53
May, 2004	$1,878,317.51
June, 2004	$1,949,361.11
July, 2004	$50,840.63

Example: **Sorting By an Alphanumeric Date Containing Calendar Quarter**

The following request against the Adventure Works cube sorts by the Calendar_Quarter field:

```
TABLE FILE adventure_works_mixed_case
SUM
Internet_Sales_Amount
BY Calendar_Quarter
WHERE Calendar_Year EQ 'CY 2001' OR 'CY 2004'
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET HTMLCSS ON
END
```

The default synonym has the following declaration for the Calendar_Quarter field:

```
FIELDNAME=Calendar_Quarter, ALIAS='Calendar Quarter', USAGE=A10,
  ACTUAL=A10,
  TITLE='Calendar Quarter',
  WITHIN='Calendar_Semester',
  PROPERTY=CAPTION,$
```

The output shows that the quarters are sorted in alphabetical order, not quarter and year order:

Calendar Quarter	Internet Sales Amount
Q1 CY 2004	$4,283,629.96
Q2 CY 2004	$5,436,429.15
Q3 CY 2001	$1,453,522.88
Q3 CY 2004	$50,840.63
Q4 CY 2001	$1,812,850.77

Now, make the following changes to the declaration for the Calendar_Quarter field:

- Change the USAGE to a WebFOCUS date format containing the quarter and year (QYY).
- Add a DATEPATTERN attribute that describes the alphanumeric date stored in the field:
  - Literal value, Q: Q
  - Variable Quarter: [Q]
  - Literal value, space followed by CY followed by space: CY
  - Variable four-digit year: [YYYY]

```
FIELDNAME=Calendar_Quarter, ALIAS='Calendar Quarter', USAGE=QYY,
  ACTUAL=A10,
  TITLE='Calendar Quarter',
  WITHIN='Calendar_Semester',
  PROPERTY=CAPTION,
  DATEPATTERN = 'Q[Q] CY [YYYY]',$
```

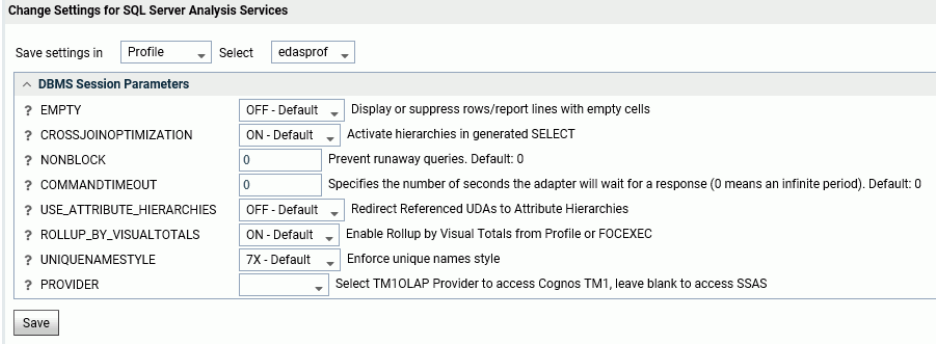
Re-running the request with these changes produces the following output in which the quarters are sorted correctly. Note that the displayed value uses the WebFOCUS date format, not the original alphanumeric date format:

Calendar Quarter	Internet Sales Amount
Q3 2001	$1,453,522.88
Q4 2001	$1,812,850.77
Q1 2004	$4,283,629.96
Q2 2004	$5,436,429.15
Q3 2004	$50,840.63

Customizing the SQL Server Analysis Services Environment

The Adapter for SQL Server Analysis Services provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

You can change these settings from the Web Console on the Adapters tab, right-clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens, as shown in the following image.



You can change the following parameters.

EMPTY

Displays or suppresses rows, or report lines, where all the selected measure columns have empty values. OFF displays the empty values. ON suppresses empty values. OFF is the default value.

CROSSJOINOPTIMIZATION

ON Optimizes queries by generating an MDX statement with CROSSJOINs of the sets of members of referenced dimensions on the ROWS axis instead of generating an MDX statement with each dimension on a separate axis. OFF suppresses CROSSJOIN optimization. This is provided for performance comparison and tuning purposes. ON is the default value.

When the EMPTY parameter is OFF (the default), this technique reduces the amount of data passed from the SQL Analysis Services client to the adapter by delegating the screening of empty tuples of data to the Analysis Services engine. The reduction is especially dramatic when low dimension levels on many dimensions are involved. (If EMPTY is ON, this option will not reduce the amount of data passed from the client to the adapter.)

NONBLOCK

NONBLOCK prevents runaway queries by specifying the polling period in seconds. This enables the user to cancel the request if necessary. The default value is 0. The Adapter for SQL Server Analysis Services uses non-blocking protocol for query execution.

COMMANDTIMEOUT

Specifies the number of seconds the adapter will wait for a response after it issues a request to SQL Server Analysis Services. The default value is 30. The value 0 represents an unlimited wait for a response.

Note: If you do not specify a COMMANDTIMEOUT value, the current SQL Server Analysis Services default timeout setting is used.

USE_ATTRIBUTE_HIERARCHIES

Controls whether the adapter automatically interprets references to UDA fields as references to the corresponding attribute hierarchies. OFF includes fields only for those hierarchies flagged as user-defined. ON includes fields for all hierarchies. OFF is the default value.

ROLLUP_BY_VISUALTOTALS

Controls whether values from the cube are recalculated so that the displayed values are determined by the report selection criteria, or are displayed as stored in the cube. ON uses the MDX VISUALTOTALS function to recalculate measure values based on the selection criteria, OFF displays the values as stored in the cube. ON is the default value.

All hierarchies within all synonyms inherit the default ROLLUP_BY_VISUALTOTALS value set in the server profile. However, in the Synonym Editor, you right-click a synonym, dimension or hierarchy to set a ROLLUP_BY_VISUALTOTALS value for:

- All hierarchies within all dimensions in a selected synonym.
- All hierarchies within selected dimensions in a selected synonym.
- Specific hierarchies within selected dimensions in a selected synonym.

UNIQUENAMESTYLE

Controls the MDX unique names style for SSAS and allows users to choose the way of forming member unique names. Valid values are 7X, NAMEPATH, or KEYPATH. The default value is 7X.

PROVIDER

Select TM1OLAP to access Cognos TM1 data. Leave blank otherwise.

Enabling Automatic Recognition of Date Patterns

The following SET command controls whether the adapter scans dimension members and automatically recognizes date and time patterns in the data. Recognized date/time patterns are described with USAGE formats and DATEPATTERN values for the corresponding fields in the generated synonym. The syntax is:

```
ENGINE SSAS SET DATEPATTERN_SCAN {ALL|DT|OFF}
```

where:

`ALL`
Specifies scanning data for all of the cube hierarchy levels.

`DT`
Specifies scanning for attributes with date- or time-related types (assigned using SQL Server Data Tools).

`OFF`
Specifies no scanning. This is the default behavior compatible with prior releases of the adapter.

Adapter Functionality

The Adapter for SQL Server Analysis Services (SSAS) supports:

- **JOIN commands.** The Adapter for SQL Server Analysis Services (SSAS) supports direct JOINs from or to a cube.

- **SUM and PRINT commands with Measures dimensions.** When using these commands in a request, the verb used against the Adapter for SQL Server Analysis Services (SSAS) does not affect the output, which is always the value found in the cube.

Example: **Using SUM and PRINT Commands in a Report Request**

In the following request, the PRINT verb is used, and SUM is the aggregator for Store_Cost. However, the value displayed is always the value stored in the cube or the specified measure at the intersection of the dimension values, regardless of the verb used in the request.

```
TABLE FILE sales_mixed_case
PRINT Store_Cost
BY City
END
```

The correct request produces the same report output:

```
TABLE FILE sales_mixed_case
SUM Store_Cost
BY City
END
```

The partial output is:

```
City          Store Cost
----          ----------
Albany        5,593.28
Altadena      2,239.71
Anacortes     641.93
Arcadia       2,045.73
Ballard       2,169.51
```

SQL Server Analysis Services (SSAS) Reporting With WebFOCUS

Two types of hierarchy are represented in a synonym: level and parent/child:

- A synonym describes a level hierarchy by using a separate field for each level. To report on a level hierarchy, you use columnar reporting in which you specify the field name for each level you want to display as a BY field in the request.
- A synonym describes a parent/child hierarchy using a set of fields that define the hierarchical structure and the relationships between the hierarchy members. The adapter has special hierarchical reporting syntax for reporting on parent/child hierarchies.

Overview of Reporting Concepts

In a multi-dimensional data source (cube), dimensions are categories of data, such as Region or Time, that you use to analyze and compare business performance. Dimensions consist of data elements that are called *members*. For example, a Region dimension could have members *England* and *France*.

Dimension members are usually organized into hierarchies. Hierarchies can be viewed as tree-like graphs where members are the nodes.

For example, the Region dimension may have the element *World* at its top level (the root node). The World element may have children nodes (members) representing continents. Continents, in turn, can have children nodes that represent countries, and countries can have children nodes representing states or cities. Nodes with no children are called *leaf nodes*.

Measures are numeric values, such as Sales Volume or Net Income, that are used to quantify how your business is performing.

A multi-dimensional cube consists of data derived from facts, which are records about individual business transactions. For example, an individual fact record reflects a sales transaction of a certain number of items of a certain product at a certain price, which occurred in a certain store at a certain moment of time. The cube contains summarized fact values for all combinations of measures and members of different dimensions.

For example, the following combination (*tuple*) contains the total volume of sales of pumps in all stores in England in 2005:

```
{Sales Volume, Pumps, England, 2005}
```

The point in the cube that contains this summarized value is called a cell. A cell is addressed by a combination of members of different dimensions and a measure. In this example *Sales Volume* is a measure and *Pumps*, *England*, and *2005* are members of the Product, Region, and Time dimensions respectively.

Individual fact records are usually tied to the leaf nodes of each hierarchy in the cube. The fact values get included in cells addressed by these leaf nodes and added to all cells addressed by all combinations of ascendants of these leaf nodes along each hierarchy of the cube.

The operation used to summarize facts for some measures can be a simple sum or a more complex aggregation function such as an average.

Some combinations of hierarchy nodes may not have any fact records tied to them. The cells addressed by these combinations are empty cells.

As illustrated in the previous example, a tuple is a combination of members from different dimensions of a cube. The previous tuple contains members from all dimensions and, therefore, addresses a single cell. If a tuple contains only members from some dimensions, it addresses not just one cell but a whole slice of cells in the cube. For example, the following tuple does not include either Region or Time dimension members:

```
{Sales Volume, Pumps}
```

It addresses as many cells in the cube as there are members of the Region dimension times the number of members of the Time dimension.

The number of cells in the cube addressed by a single dimension member is a product of cardinalities of all other dimensions.

When all cells addressed by a member are empty, the member is called an empty member. When all cells addressed by a tuple are empty, the tuple is called an empty tuple.

By default, empty cells do not appear on report output. You can issue the following command if you want them on the report output:

```
ENGINE SSAS SET EMPTY ON
```

OLE DB for OLAP (ODBO) defines three basic types of hierarchy structures: balanced, unbalanced, and ragged.

- In balanced hierarchies, all leaf members belong to the lowest hierarchy level. If a hierarchy member has a parent, the parent is exactly one level above the member.
- In unbalanced hierarchies, not all leaf members belong to the lowest level of the hierarchy.
- In ragged hierarchies, some members have their immediate parents (children) more than one level above (below) them.

The SSAS implementation of OLAP introduces its own terminology. SSAS hierarchies can be standard or parent-child. This characteristic has more to do with the way the hierarchy is defined rather than with its actual structure. Standard hierarchies are created from relational tables in such a way that each level of the hierarchy corresponds to a column in a relational table (of a column-based expression). Parent-child hierarchies, however, are created from tables having two essential columns. One containing the unique identifier of a member and another containing the identifier of its parent.

Thus, for standard hierarchies, the number of levels is fixed at the time when the hierarchy is defined. For parent-child hierarchies, the number of levels is determined by the contents of tables from which they are created rather than their columnar structure. Therefore, this number may change each time the hierarchy is loaded.

Usually, names of levels of standard hierarchies are inherited from the names of the corresponding relational columns because the data in different columns of a relational table are organized according to its semantics. Data items with the same semantics belong to the same column (for example country names belong to the *Country* column, city names belong to the *City* column). Level names of parent-child hierarchies are either automatically generated or assigned by the OLAP administrator, depending on the nature of the hierarchy data because data items in parent-child tables tend to be homogeneous. A list of company employees is a typical example of data that is naturally organized this way.

At the time of creation of a standard hierarchy it is possible to specify the convention for missing members. For example, a NULL value might represent a missing member. For example, a NULL value in the State column in a row of the Geography table containing values 'USA'-NULL-'Washington DC' means that the Washington DC member's parent is USA, which is two levels above it. This mechanism allows creation of standard hierarchies which, from the ODBO point of view, have unbalanced and/or ragged structures.

For parent-child hierarchies, SSAS currently does not provide an option for specifying a missing member. Thus it is currently impossible to create a parent-child hierarchy with a ragged structure. However, parent-child hierarchies typically (but not necessarily) have an unbalanced structure.

This difference between standard and parent-child hierarchies also explains why different levels of a standard hierarchy usually have different sets of user-defined properties, while all levels of a parent-child hierarchy have identical sets of user-defined properties.

When the adapter accesses a multi-dimensional cube, it uses two types of metadata elements about dimensions:

- **Hierarchy fields.** These fields contain data that apply to a specific hierarchy and describe each member position within that hierarchy. For example, these fields identify the member parent as well as its own name, caption, and unique ID.
- **Dimension properties.** These fields contain data that potentially apply to all members of the dimension. For example, a Region dimension may have a property called GEOGRAPHICAL_HEIGHT that specifies the altitude for each member of the Region dimension.

Using the hierarchy fields, the adapter can recreate the hierarchy and locate portions of the hierarchy needed to satisfy a request.

Understanding Columnar and Hierarchical Reporting

Hierarchical reporting enables you to sort and select members of parent/child hierarchies without knowing specific level numbers.

A hierarchical reporting request goes through several phases before output is displayed.

Hierarchical Sorting and Member Selection

The first phase selects hierarchy members to display. The hierarchical reporting phrase BY or ON HIERARCHY automatically sorts and formats a hierarchy with appropriate indentations that show the parent/child relationships. If you do not want to see the entire hierarchy, you can use the WHEN phrase to select hierarchy members for display. The expression in this WHEN phrase must reference only hierarchy fields, not dimension properties or measures.

Measures cannot be used in selecting hierarchy levels for display.

Screening Dimension Data

Members are selected using the WHEN phrase. WHERE and IF phrases are not supported.

Screening Based on Aggregated Values

Measures, being summarized values, can be referenced in WHERE TOTAL tests and COMPUTE commands because those commands are processed after the hierarchy selection and aggregation phases of the request. When screening with WHERE TOTAL, the aggregation phase of the report processing is over, so totals on the report are not recalculated to account for the data that is screened out, the rows are just removed.

Reference: Prerequisites for Hierarchical Reporting

Hierarchical reporting uses special metadata attributes and reporting syntax. You must:

- Create a synonym that describes hierarchies as parent/child relationships.
- Use hierarchical reporting syntax in your request to automatically build and format hierarchies in the report output.

Representing Hierarchies in a Synonym

Dimensions are organized into sets of hierarchies. For example, in a Time dimension, years, quarters, and months can form a hierarchical or parent/child relationship. This means that the measures for each month are aggregated into values for quarters, and the quarters are aggregated into values for years. Each point in the hierarchy is called a *node*. Nodes at the bottom of the hierarchy (with no children) are called *leaf nodes*.

In a synonym, hierarchies can be described in one of two ways:

- **Level hierarchy.** Each hierarchy level is described using a separate field name. To issue a report request, you must reference the field name for those levels you want to display on the report output.

When you create a synonym, the synonym contains one field declaration for each level. This declaration also specifies which field is its parent. If the data changes to have an additional level, you must recreate the synonym in order to account for this additional field and parent reference.

- **Parent/child hierarchy.** The hierarchy is described with a set of fields that contain values for properties that describe each member position in the hierarchy. For example, there are fields to contain a member unique ID, parent, and caption. To issue a report request, you only need to specify the field name of one of the hierarchy fields.

  With a parent/child hierarchy, one set of field names in the synonym describes the hierarchy. A change in the number of levels does not require a change to the synonym.

  Another advantage of parent/child hierarchies is that the adapter can recreate and format any portion of the hierarchy. The request does not have to specify level numbers.

Dimension properties apply to all hierarchies in the dimension and are listed in the synonym following all of the hierarchies.

Example: Dimension Declaration

Each dimension begins with a dimension record that defines the dimension and its hierarchies. The dimension itself is level zero.

Dimension:

```
DIMENSION=[Employee], CAPTION='Employee', $
```

Example: Describing a Level Hierarchy

Each level of the hierarchy is assigned a field name consisting of the hierarchy name with the level number appended. Each field declaration also specifies the field name of its parent with the WITHIN attribute. The value stored in this field is the member caption (title).

If a new level appears in the data, the synonym must be recreated to define this new level.

The following field describes level three. Its parent is level 2:

```
FIELDNAME=Product_Category, ALIAS='Product Category', USAGE=A20,
     ACTUAL=A20,
     TITLE='Product Category',
     WITHIN='Product_Department',
     PROPERTY=CAPTION,  $
```

Example: Describing a Parent/Child Hierarchy

Several fields are used to define a parent/child hierarchy. Each has a PROPERTY attribute that describes which hierarchy property it represents. The hierarchy field names are formed by appending a suffix to the hierarchy name.

For example, the caption of a hierarchy named Departments is stored in a field whose name is Departments and whose property attribute is PROPERTY=CAPTION.

The following table describes the hierarchy fields:

Description of Data	PROPERTY=	Field Suffix
Member's Unique ID (unique within the cube)	UID	
Member's Name (unique within the hierarchy)	NAME	_name
Member's Level Number	LEVEL_NUMBER	_lvlno
Member's Parent	PARENT_OF	_parent
Parent's Level Number	PARENT_LEVEL_NUMBER	_parent_lvlno
Number of Children	CHILDREN_CARDINALITY	_children_card
Member's Caption (title on reports)	CAPTION	_caption

The following declaration for the Departments hierarchy describes the field that contains a member's unique ID (PROPERTY=UID):

```
FIELDNAME=Departments, USAGE=A143, ACTUAL=A143,
     MISSING=ON,
     TITLE='Departments Member Unique Name',
     WITHIN='*[Department].[Departments]',
     REFERENCE=[Department], PROPERTY=UID,  $
```

The following declaration for the Departments hierarchy defines the field that contains a member's title (PROPERTY=CAPTION):

```
   FIELDNAME=Departments_caption, USAGE=A60, ACTUAL=A60,
     MISSING=ON,
     TITLE='Departments Member Caption',
     REFERENCE=ADVENTURE_WORKS.Departments, PROPERTY=CAPTION
```

Example: Dimension Properties

Following all hierarchies, the dimension properties (called *attributes*) are described. Each of these has PROPERTY=ATTRIBUTE in the synonym.

For example, the following field represents the property *occupation* in the Customers hierarchy:

```
 FIELDNAME=Occupation, ALIAS=Occupation, USAGE=A14, ACTUAL=A14,
      MISSING=ON,
      TITLE='Occupation',
      REFERENCE=Customer, PROPERTY=ATTRIBUTE,  $
```

Example: Sample Request Using a Level Hierarchy

A report request against a level hierarchy must specify the field name for each level of the hierarchy required in the report.

For example, the following request displays the Profit measure for levels 1 through 3 of the Customers hierarchy:

```
TABLE FILE sales_mixed_case
SUM
 Profit
 BY Country AS 'Level1'
 BY State_Province AS 'Level2'
 BY City AS 'Level3'
ON TABLE SET PAGE NOLEAD
END
```

The partial output is:

Level1	Level2	Level3	Profit
USA	CA	Altadena	3,345.88
		Arcadia	3,090.86
		Bellflower	3,995.50
		Berkeley	190.88
		Beverly Hills	3,714.91
		Burbank	3,954.29
		Burlingame	246.96

In order to get a total for the entire hierarchy, you have to use an ON TABLE COLUMN-TOTAL command in the request or add another SUM command without a BY phrase (and this would add another column to the report output).

Example: Sample Request Using a Parent/Child Hierarchy

A report request against a parent/child hierarchy can use the BY HIERARCHY phrase to report against the entire hierarchy. The output is automatically formatted with appropriate indentations to show the hierarchy levels and relationships.

For example, the following request shows the sales amount measure for three generations of the Organizations hierarchy:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
SHOW TO DOWN 3
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The partial output is:

```
Accounts Member Caption                Amount
-----------------------                ------
  Balance Sheet                           .00
    Assets                        13,740,731.00
      Current Assets              12,445,628.00
        Cash                       3,236,799.00
        Receivables                3,475,923.00
          Trade Receivables        3,371,580.00
          Other Receivables          104,343.00
        Allowance for Bad Debt        67,429.00
        Inventory                  4,143,398.00
          Raw Materials            2,007,586.00
          Work in Process          1,393,582.00
          Finished Goods             742,230.00
        Deferred Taxes               505,424.00
        Prepaid Expenses             341,992.00
```

The report request does not have to reference specific hierarchy levels. The BY HIERARCHY phrase recreates and formats the hierarchy for display. You can also use a WHEN phrase to select a portion of the hierarchy and a SHOW phrase to specify how many levels above and below the selected portion of the hierarchy you want to display. This display format clearly shows the parent/child relationships between the hierarchy members.

For more information, see *Hierarchical Reporting* on page 1484.

Hierarchical Reporting

When you issue a request against a cube, some of the requirements and features available depend on the type of hierarchy you are reporting against.

With a parent/child hierarchy, you can specify whether the measure values displayed for each parent should show the sum of all of its descendants (full total) or the sum of its displayed descendants (visual total).

Rollup using the MDX VISUALTOTALS function is a SSAS Adapter feature that enables you to select members using WHERE/IF clauses (for level hierarchies) or WHEN clauses (for parent/child hierarchies), and have the SSAS engine recalculate the values for the displayed members based on the report selection criteria. MDX is the language that is used by the SSAS engine.

For parent/child hierarchies, you can use the BY HIERARCHY phrase to sort and format the hierarchy. You can also limit the portion of the hierarchy selected for display using the WHEN phrase.

When a hierarchical request is processed, the first step is to build the hierarchy and mark which nodes should be included, which should be excluded, and which are needed for context.

The next stage fills the hierarchy with measure values.

Measure values cannot be used to select hierarchy levels for reporting. After the hierarchy rows have been selected, screened, and aggregated, WHERE TOTAL tests can limit the rows displayed based on measure values.

Syntax: How to Display Parent/Child Hierarchies

The following syntax can be used to generate hierarchical reports when the synonym defines parent/child hierarchies:

```
SUM measure_field ...
BY hierarchy_field [HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO {BOTTOM|DOWN m}] [byoption [WHEN condition] ...] ]
.
.
.
[ON hierarchy_field HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO BOTTOM|DOWN m] [byoption [WHEN condition] ...]]
```

where:

measure_field

Is the field name of a measure.

BY *hierarchy_field* HIERARCHY

Identifies the hierarchy used for sorting.

`ON` *`hierarchy_field`* `HIERARCHY`

Identifies the hierarchy used for sorting. The request must include either a BY phrase or a BY HIERARCHY phrase for this field name.

`WHEN` *`expression_using_hierarchy_fields`*;

Selects hierarchy members. The WHEN phrase must immediately follow the word HIERARCHY to distinguish it from a WHEN phrase associated with a BY option (such as SUBFOOT). Any expression using only hierarchy fields or properties is supported. The WHEN phrase can be on the BY HIERARCHY command or the ON HIERARCHY command, but not both.

`SHOW`

Specifies which levels to show on the report output relative to the levels selected by the WHEN phrase. If there is no WHEN phrase, the SHOW option is applied to the root node of the hierarchy. The SHOW option can be specified on the BY HIERARCHY phrase or the ON HIERARCHY phrase, but not both.

`n`

Is the number of ascendants above the set of selected members that will have measure values. All ascendants appear on the report to show the hierarchical context of the selected members. However, ascendants that are not included in the SHOW phrase appear on the report with missing data symbols in the report columns that display measures. The default for *n* is 0.

`TOP`

Specifies that ascendant levels to the root node of the hierarchy will be populated with measure values.

`TO`

Is required when specifying a SHOW option for descendant levels.

`BOTTOM`

Specifies all descendants to the leaf nodes of the hierarchy will be populated with measure values. This is the default value.

`m`

Is the number of descendants of each selected level that will display. The default for *m* is BOTTOM, which displays all descendants.

byoption

Is one of the following sort-based options: PAGE-BREAK, REPAGE, RECAP, RECOMPUTE, SKIP-LINE, SUBFOOT, SUBHEAD, SUBTOTAL, SUB-TOTAL, SUMMARIZE, UNDER-LINE. If you specify SUBHEAD or SUBFOOT, you must place the WHEN phrase on the line following the heading or footing text.

condition

Is a logical expression.

Note: To generate a visual total (sum that represents only the displayed members) rather than a full total (value found in the cube), set the ROLLUP_BY_VISUALTOTALS parameter in the Web Console.

The following examples illustrate hierarchical reporting using the BY HIERARCHY phrase. Note that requests with multiple display commands are not supported.

For information about using the MDX VISUALTOTALS function, see *Customizing the SQL Server Analysis Services Environment* on page 1472 and *Effect of MDX ROLLUP_BY_VISUALTOTALS Mode on Report Output* on page 1497.

Example: Reporting on a Whole Hierarchy

The following request displays the Amount measure for the Accounts hierarchy. The BY HIERARCHY phrase specifies hierarchical reporting. There is no WHEN phrase to limit the portion of the hierarchy displayed:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Reporting on a Whole Hierarchy"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The top portion of the hierarchy is:

Reporting on a Whole Hierarchy

Accounts Member Caption	Amount
Balance Sheet	$.00
Assets	$13,740,731.00
Current Assets	$12,445,628.00
Cash	$3,236,799.00
Receivables	$3,475,923.00
Trade Receivables	$3,371,580.00
Other Receivables	$104,343.00
Allowance for Bad Debt	$67,429.00
Inventory	$4,143,398.00
Raw Materials	$2,007,586.00
Work in Process	$1,393,582.00
Finished Goods	$742,230.00
Deferred Taxes	$505,424.00
Prepaid Expenses	$341,992.00
Intercompany Receivables	$674,663.00
Property, Plant, Equipment	$1,122,394.00
Land & Improvements	$93,843.00
Buildings & Improvements	$299,043.00
Machinery & Equipment	$92,606.00
Office Furniture & Equipment	$523,016.00
Leasehold Improvements	$80,759.00
Construction In Progress	$33,127.00
Other Assets	$172,709.00
Liabilities and Owners Equity	$13,740,731.00
Liabilities	$5,664,258.00
Current Liabilities	$3,738,221.00
Notes Payable	$148,316.00
Current Installments of Long-term Debt	$149,571.00
Accounts Payable	$1,286,664.00
Accrued Expenses	$865,491.00

Example: Selecting a Hierarchy Member

The following request displays the Amount measure for the Accounts dimension, but limits the member selected for display with the WHEN phrase. Note that the hierarchy member selected is displayed in its context (all ancestors to the root of the hierarchy). However, the ancestors are not requested in the report and are, therefore, displayed with missing data symbols. All descendants of the selected members appear in the report output because the default SHOW option for descendants is BOTTOM:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Accounts_caption CONTAINS 'Receivables';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Hierarchy Member"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Hierarchy Member

Accounts Member Caption	Amount
Balance Sheet	.
Assets	.
Current Assets	.
Receivables	3,475,923.00
Trade Receivables	3,371,580.00
Other Receivables	104,343.00
Intercompany Receivables	674,663.00

Example: Selecting a Member and Adding a Parent

In the following request, the SHOW option UP 1 TO DOWN 0 added to the WHEN phrase adds the parent (Current Assets) of the selected member (Receivables). This parent now contains values for the measure rather than a missing data symbol:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Accounts_caption EQ 'Receivables';
SHOW UP 1 TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding a Parent"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding a Parent

Accounts Member Caption	Amount
Balance Sheet	.
Assets	.
Current Assets	3,475,923.00
Receivables	3,475,923.00

Example: **Selecting a Member and Adding Children**

In the following request, the SHOW option UP 0 TO DOWN 1 added to the WHEN phrase adds the children of the selected member (Receivables):

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Accounts_caption EQ 'Receivables';
SHOW UP 0 TO DOWN 1
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding Children"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding Children

Accounts Member Caption	Amount
Balance Sheet	.
Assets	.
Current Assets	.
Receivables	3,475,923.00
Trade Receivables	3,371,580.00
Other Receivables	104,343.00

Example: Selecting a Member and Showing All Ascendants

In the following request, the SHOW option TOP TO DOWN 0 added to the WHEN phrase adds all ascendants but no descendants of the selected member (Receivables). These parents now contain values for the measure rather than missing data symbols:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Accounts_caption EQ 'Receivables';
SHOW TOP TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding All Ascendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding All Ascendants

Accounts Member Caption	Amount
Balance Sheet	($10,264,808.00)
Assets	$3,475,923.00
Current Assets	$3,475,923.00
Receivables	$3,475,923.00

Example: Selecting a Member and Showing All Ascendants and Descendants

In the following request, the SHOW option TOP added to the WHEN phrase adds all ascendants and descendants (since TO BOTTOM is the default) of the selected member (Receivables). These parents are now in the SHOW set and contain values for the measure rather than missing data symbols:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Accounts_caption EQ 'Receivables';
SHOW TOP
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding All Ascendants and Descendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding All Ascendants and Descendants

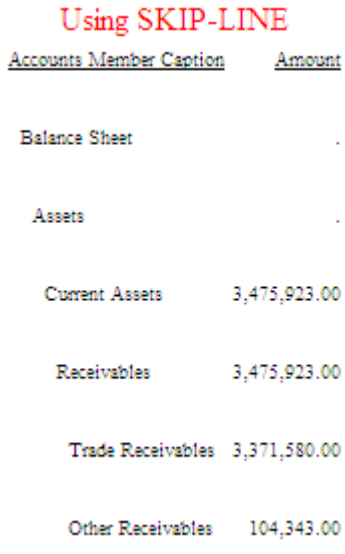
Accounts Member Caption	Amount
Balance Sheet	($10,264,808.00)
Assets	$3,475,923.00
Current Assets	$3,475,923.00
Receivables	$3,475,923.00
Trade Receivables	$3,371,580.00
Other Receivables	$104,343.00

Example: Using SKIP-LINE

The following request adds the BY option SKIP-LINE to the BY HIERARCHY phrase:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Accounts_caption EQ 'Receivables';
SHOW UP 1 TO DOWN 1 SKIP-LINE
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using SKIP-LINE"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:



Using SKIP-LINE

Accounts Member Caption	Amount
Balance Sheet	.
Assets	.
Current Assets	3,475,923.00
Receivables	3,475,923.00
Trade Receivables	3,371,580.00
Other Receivables	104,343.00

Example: Using Conditional Sort Options

The following request has a BY HIERARCHY command with a WHEN phrase to select members as well as a WHEN phrase to control the UNDER-LINE option.

The SUBFOOT option is in an ON phrase that references the same hierarchy field (all of the BY options could have been on the BY HIERARCHY phrase).

Each BY option has its own WHEN phrase. The WHEN phrase for the SUBFOOT option uses a measure field in its expression. The WHEN phrase for the UNDER-LINE option uses a hierarchy field in its expression. The WHEN phrase that selects members uses a dimension property in its expression.

In this example, one BY option is activated WHEN the Accounts_caption contains *Receivables*. The other BY option creates a subfoot WHEN Amount is greater than or equal to $4,000,000:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption HIERARCHY
WHEN Account_Type EQ 'Assets';
SHOW TOP UNDER-LINE WHEN Accounts_caption CONTAINS 'Receivables';
ON Accounts_caption SUBFOOT
" "
"The Assets are Large"
" "
WHEN Amount GE 4000000;
ON TABLE SUBHEAD
"Using BY Options With WHEN on a Measure and a Hierarchy Field"
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using BY Options With WHEN on a Measure and a Hierarchy Field

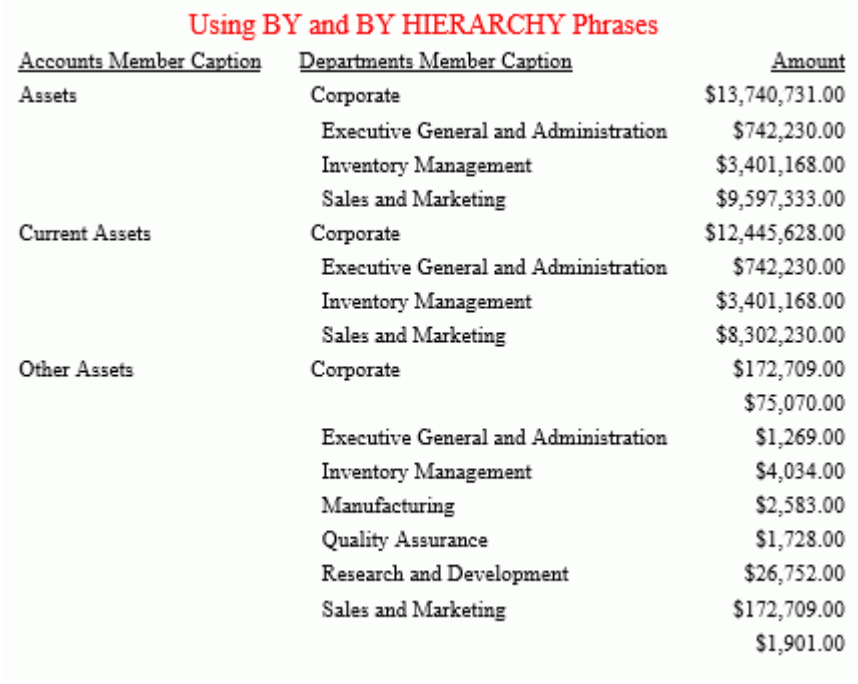
Accounts Member Caption	Amount
Balance Sheet	$.00
Assets	$13,740,731.00
The Assets are Large	
Current Assets	$12,445,628.00
The Assets are Large	
Cash	$3,236,799.00
Receivables	$3,475,923.00
Trade Receivables	$3,371,580.00
Other Receivables	$104,343.00
Allowance for Bad Debt	$67,429.00
Inventory	$4,143,398.00
The Assets are Large	
Raw Materials	$2,007,586.00
Work in Process	$1,393,582.00
Finished Goods	$742,230.00
Deferred Taxes	$505,424.00
Prepaid Expenses	$341,992.00
Intercompany Receivables	$674,663.00
Property, Plant, Equipment	$1,122,394.00
Land & Improvements	$93,843.00
Buildings & Improvements	$299,043.00
Machinery & Equipment	$92,606.00
Office Furniture & Equipment	$523,016.00
Leasehold Improvements	$80,759.00
Construction In Progress	$33,127.00
Other Assets	$172,709.00

Example: Using BY HIERARCHY and BY Phrases

The following request has a BY phrase for the Accounts hierarchy and a BY HIERARCHY phrase for the Departments hierarchy. All selected members for the Departments hierarchy are repeated for each selected member of the Accounts hierarchy (which is not displayed with hierarchical indentations when referenced in a BY phrase). A BY on a unique field is required. WHEN is used to select members for the BY HIERARCHY phrase. WHERE is used to select rows for the BY phrase:

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Accounts_caption
BY Departments_caption HIERARCHY
WHEN Departments_caption OMITS 'Sales';
WHERE Accounts_caption CONTAINS 'Assets'
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using BY and BY HIERARCHY Phrases"
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:



Using BY and BY HIERARCHY Phrases

Accounts Member Caption	Departments Member Caption	Amount
Assets	Corporate	$13,740,731.00
	Executive General and Administration	$742,230.00
	Inventory Management	$3,401,168.00
	Sales and Marketing	$9,597,333.00
Current Assets	Corporate	$12,445,628.00
	Executive General and Administration	$742,230.00
	Inventory Management	$3,401,168.00
	Sales and Marketing	$8,302,230.00
Other Assets	Corporate	$172,709.00
		$75,070.00
	Executive General and Administration	$1,269.00
	Inventory Management	$4,034.00
	Manufacturing	$2,583.00
	Quality Assurance	$1,728.00
	Research and Development	$26,752.00
	Sales and Marketing	$172,709.00
		$1,901.00

Example: **Using ON HIERARCHY Without BY HIERARCHY**

The following request has a BY phrase and an ON HIERARCHY phrase for the Departments hierarchy. All hierarchy options and BY options are supported on the ON HIERARCHY phrase. This request also has a WHEN clause that selects members based on the value of a hierarchy field (Departments_caption):

```
TABLE FILE adventure_works_mixed_case
SUM Amount
BY Departments_caption
ON Departments_caption HIERARCHY
WHEN Departments_caption OMITS 'Sales';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using ON HIERARCHY"
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using ON HIERARCHY

Departments Member Caption	Amount
Corporate	$12,609,503.00
Executive General and Administration	($396,702.00)
Inventory Management	($1,315,394.00)
Manufacturing	($826,648.00)
Quality Assurance	($574,566.00)
Research and Development	$28,625,144.00
Sales and Marketing	($639,841.00)

Effect of MDX ROLLUP_BY_VISUALTOTALS Mode on Report Output

When reporting against SQL Server Analysis Services (SSAS) cubes, by default, the report output displays the visual totals, not the measure values stored in the cube cells for members that are displayed on the report output. The values displayed are determined by the report selection criteria and dimension levels referenced in the sort phrases (BY and ACROSS) for level hierarchies, or explicit member selection for Parent/Child hierarchies and dimensions not referenced by sort phrases.

To retrieve data from a SSAS cube, the adapter translates the WebFOCUS request to MDX, the language that is used by the SSAS engine. By default, the adapter uses the MDX VISUALTOTALS function. SSAS then returns the requested cells with visual totals.

You can set ROLLUP_BY_VISUALTOTALS OFF if you want values that are aggregates resulting from rolling up measures along the cube dimensions when the cube was processed, and are based on rolling up all the underlying members of the hierarchy even if the report performs some selection on the levels below the displayed ones.

Rollup using the MDX VISUALTOTALS function is a SSAS Adapter feature that enables you to select members using WHERE/IF clauses (for level hierarchies) or WHEN clauses (for parent/child hierarchies), and have the SSAS engine recalculate the values for the displayed members based on the report selection criteria.

To set the ROLLUP_BY_VISUALTOTALS parameter, see *Customizing the SQL Server Analysis Services Environment* on page 1472.

Reporting Against Level Hierarchies

Level hierarchies are presented in WebFOCUS synonyms as sets of fields consisting of one mandatory field per hierarchy level (for member captions) and a field per a user-defined property (attribute relationship) associated with the given level. The adapter employs the VISUALTOTALS function when the hierarchy levels referenced in filter conditions (WHERE, IF clauses) are below the levels selected for sort/grouping (BY clauses). The set of members to which the VISUALTOTALS function is applied consists of all the filtered members and their hierarchical ascendants up to (but not above) the lowest level selected for sort/grouping.

Example: **Level Hierarchies With Sort/Grouping**

Consider the following hierarchy flattened into a table:

Country	State/Province	City
Australia	**New South Wales**	**Coffs Harbour**
Australia	New South Wales	**Darlinghurst**
Australia	New South Wales	**Goulburn**
Australia	**Victoria**	**Melbourne**
Australia	Victoria	**Sunbury**
Canada	**British Columbia**	**Burnaby**
Canada	British Columbia	**Cliffside**
United States	**California**	**Los Angeles**
United States	California	**San Francisco**
United States	**Oregon**	**Lebanon**
United States	Oregon	**Portland**
United States	**Washington**	**Bellevue**
United States	Washington	**Seattle**
United States	Washington	**Spokane**

Each row in the table corresponds to a hierarchical path. The following WebFOCUS query selects some paths from the hierarchy using a WHERE clause:

```
TABLE FILE adventure_works_upper_case
WRITE INTERNET_SALES_AMOUNT
BY COUNTRY
WHERE CITY EQ 'Coffs Harbour' OR 'Darlinghurst' OR 'Melbourne'
OR 'Portland' OR 'Seattle' OR 'Spokane';
END
```

The following hierarchical paths are selected:

Country	State/Province	City
Australia	**New South Wales**	**Coffs Harbour**
Australia	New South Wales	**Darlinghurst**
Australia	**Victoria**	**Melbourne**
United States	**Oregon**	**Portland**
United States	**Washington**	**Seattle**
United States	Washington	**Spokane**

The BY COUNTRY phrase determines that members for reporting are selected on the level of Country. The following MDX query is a simplified version of the actual MDX generated by the adapter when ROLLUP_BY_VISUALTOTALS is set to OFF:

```
Select {[Internet Sales Amount]} on Axis(0),
{[Australia], [United States]} on Axis(1) From [Adventure Works]
```

The report has the following contents, where the values of the measure *Internet Sales Amount* are extracted for members *Australia* and *United States* as they are rolled up in the cube, not just for the cities selected by the WHERE criteria. This is the default adapter behavior:

PAGE 1	
Country	Internet Sales Amount
Australia	$9,061,000.58
United States	$9,389,789.51

For some applications, it is desirable that the values of *Country* level members shown in the report are not the values that are stored in the cube but rather values rolled up based only on the selected members of *City* level. In this case, the *Internet Sales Amount* value of Australia would be the sum of *Internet Sales Amount* for *Coffs Harbour*, *Darlinghurst*, and *Melbourne,* and the value for United States would be the sum of values for *Portland*, *Seattle,* and *Spokane*.

To do this, the adapter generates an MDX query using the MDX VISUALTOTALS function.

Note that the six selected hierarchical paths form two subtrees. Applying the VISUALTOTALS function to the set of members that form these subtrees and then intersecting the result with the members of the *Country* level provides the values for Australia and United States rolled up based on the selected cities only. The WebFOCUS request is:

```
TABLE FILE ADVENTURE_WORKS_UPPER_CASE
WRITE INTERNET_SALES_AMOUNT
BY COUNTRY
WHERE CITY EQ 'Coffs Harbour' OR 'Darlinghurst' OR 'Melbourne'
OR 'Portland' OR 'Seattle' OR 'Spokane';
END
```

The following is a simplified version of the MDX query generated by the adapter to achieve the visual total.

```
Select {[Internet Sales Amount]} on Axis(0),
Intersect(
VisualTotals({[Australia], [New South Wales], [Coffs Harbour],
[Darlinghurst],
[Victoria], [Melbourne],
[United States],[Oregon],[Portland],
[Washington],[Seattle],[Spokane]}
), [Customer Geography].[Country].Members)
on Axis(1)
From [Adventure Works]
```

The values that result from this query are not the values for Australia and United States taken from the cube but rather values rolled up by SSAS from the selected cities:

PAGE 1	
Country	Internet Sales Amount
Australia	$637,022.87
United States	$254,920.12

Example: **Level Hierarchies With Calculated Measures**

The following example shows how visual totals work for calculated measures. The calculated measure [*Internet Average Unit Price*] is defined by the following expression:

[*Internet Unit Price*] / [*Internet Transaction Count*]

where [*Internet Unit Price*] and [*Internet Transaction Count*] are measures that are not visible in the Adventure Works cube metadata (and, therefore, the WebFOCUS synonym does not contain corresponding fields for them). Using the MDX VISUALTOTALS function, the following WebFOCUS query returns correct results for the selected countries based only on the selected cities:

```
TABLE FILE ADVENTURE_WORKS_UPPER_CASE
WRITE INTERNET_AVERAGE_UNIT_PRICE/F12.2
BY COUNTRY
WHERE CITY EQ 'Coffs Harbour' OR 'Darlinghurst' OR 'Melbourne'
OR 'Portland' OR 'Seattle' OR 'Spokane';
END
```

The output is:

PAGE 1	
Country	Internet Average Unit Price
Australia	597.02
United States	354.06

Hierarchical Reporting From Parent-Child Hierarchies

In hierarchical reports, the selected members (specified by WHEN and SHOW clauses) are always shown in their hierarchical context so that the report forms a contiguous subtree (or, if and only if the hierarchy has multiple roots, several subtrees). The values of the requested members are, by default, rolled up, not just extracted from the cube.

The exact set to which the VISUALTOTALS function is applied consists of the members selected by WHEN and SHOW clauses plus connecting members in the hierarchy. WHEN and SHOW clauses may select one or more pairs of members, one of which is an ancestor of another (and the second of which is a descendant of the first) but not the members in between. The adapter calls the VISUALTOTALS function to add all intermediate members connecting these two members to the set to which the VISUALTOTALS function is applied. This is done to ensure its proper functioning. Members below or above this connected set are not added.

Reference: Reporting From Parent-Child Hierarchies With BY and WHERE Clauses

Parent-Child hierarchies can be used for regular reporting with BY and WHERE clauses as opposed to BY HIERARCHY and WHEN. In that case, all members of a hierarchy are viewed as a flat stream without any hierarchical relationship. A report shows the selected members in a single column sorted according to the specified BY clause. In this case the notion of visual totals is not applicable.

Example: Retrieving Visual Totals in a Hierarchical Report

Consider the following WebFOCUS hierarchical reporting query when the ROLLUP_BY_VISUALTOTALS parameter is set to OFF:

```
TABLE FILE ADVENTURE_WORKS_UPPER_CASE
WRITE AMOUNT
BY ACCOUNTS_CAPTION
HIERARCHY
WHEN ACCOUNTS_CAPTION EQ 'Balance Sheet'
OR 'Assets' OR 'Liabilities and Owners Equity'
OR 'Cash'
OR 'Current Liabilities' OR 'Long Term Liabilities';
SHOW UP 0 TO DOWN 0
END
```

This results in the following report:

PAGE 1	
Accounts Member Caption	Amount
Balance Sheet	.00
Assets	13,740,731.00
Current Assets	.
Cash	3,236,799.00
Liabilities and Owners Equity	13,740,731.00
Liabilities	.
Current Liabilities	3,738,221.00
Long Term Liabilities	1,926,037.00

To show values of these members not as they are in the cube but rolled up only from its descendants shown in the same report, set the ROLLUP_BY_VISUALTOTALS parameter to ON:

```
TABLE FILE ADVENTURE_WORKS_UPPER_CASE
WRITE AMOUNT/D12.2BM
BY ACCOUNTS_CAPTION
HIERARCHY
WHEN ACCOUNTS_CAPTION EQ 'Balance Sheet'
OR 'Assets' OR 'Liabilities and Owners Equity'
OR 'Cash'
OR 'Current Liabilities' OR 'Long Term Liabilities';
SHOW UP 0 TO DOWN 0
END
```

The exact set to which the VISUALTOTALS function is applied consists of the members selected by WHEN and SHOW clauses plus connecting members. WHEN and SHOW clauses may select one or more pairs of members one of which is an ancestor of another but not the members in between (Assets and Cash in the following output). The adapter adds all intermediate members connecting these two members to the set to which the VISUALTOTALS function is applied. This is done to ensure its proper functioning. Members below or above this connected set are not added:

PAGE 1	
Accounts Member Caption	Amount
Balance Sheet	($2,427,459.00)
Assets	$3,236,799.00
Current Assets	.
Cash	$3,236,799.00
Liabilities and Owners Equity	$5,664,258.00
Liabilities	.
Current Liabilities	$3,738,221.00
Long Term Liabilities	$1,926,037.00

Reporting Without Sort/Grouping

The VISUALTOTALS function is applied to the set consisting of all ascendants of the selected members and then extracting their lowest common ancestor.

If selected members do not have common ancestors, the adapter creates one by applying function AGGREGATE to the ancestors of the selected members on the topmost level of the hierarchy. This aggregate value is then displayed on the report output.

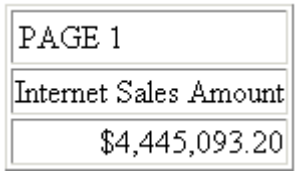
Example: Reporting Against Hierarchies Without Sort/Grouping

Consider the following WebFOCUS query:

```
TABLE FILE ADVENTURE_WORKS_UPPER_CASE
WRITE INTERNET_SALES_AMOUNT
WHERE COUNTRY EQ 'Canada' OR STATE_PROVINCE EQ 'Washington';
END
```

The default behaviour results in the combined value of both Canada and Washington members.

Since the selected members do not have common ancestors, with visual totals on, the adapter creates one by applying function AGGREGATE to the ancestors of the selected members on the topmost level of the hierarchy. This aggregate value is then displayed on the report output:

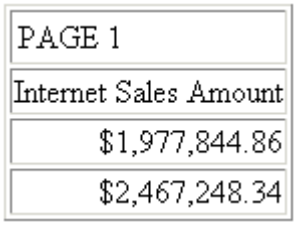


PAGE 1
Internet Sales Amount
$4,445,093.20

In this example, the generated MDX is equivalent to the following:

```
Select {[Internet Sales Amount]} on Axis(0),
Intersect(
VisualTotals({[Customer].[Customer Geography].[(All)],
[Canada],
[United States],
[Washington]})
, [Customer Geography].[(All)].Members)
on Axis(1)
From [Adventure Works]
```

Turning the ROLLUP_BY_VISUALTOTALS parameter to OFF results in the following report in which the values are shown separately for Canada and Washington:



PAGE 1
Internet Sales Amount
$1,977,844.86
$2,467,248.34

Reporting on SQL Server Analysis Services (SSAS) Sets

A *named set* is a group of members from one or more hierarchies. WebFOCUS supports named sets in which the members all come from a single hierarchy. When you create a synonym, for each hierarchy that has one or more single-hierarchy named sets, a field with a _SETS suffix is generated. The value stored in this field is the name of one or more sets defined for the associated hierarchy. A SETS field has the attribute PROPERTY=SETS.

You can find the name of a set by printing the value of its associated field. When a SETS field is used in a PRINT command, it must be the only active field in the TABLE request. Once you know the name of a set, you can use it in:

- A WHERE or IF phrase when reporting on a level hierarchy.
- A WHEN phrase when reporting on a parent/child hierarchy.

An expression referencing a named set can only consist on EQ and NE conditions. The EQ condition selects all members of the set, and the NE condition excludes all members in the set.

Note: In order to issue a request that performs a selection on a SET, the ROLLUP_BY_VISUALTOTALS parameter must be set to OFF.

Set members can come from different levels in a hierarchy. If a WHERE or SHOW condition limits the portion of the hierarchy being displayed on the report output, only the set members applicable to those hierarchy levels are shown.

Syntax: How to Select or Exclude Set Members

For columnar reporting use the following type of expression in one or more WHERE or IF clauses. For hierarchical reporting, use the following type of expression in a WHEN clause:

```
{WHERE|WHEN} field1_SETS {EQ|NE} setname11 [OR setname12 ...]
      [{AND|OR} field2_SETS {EQ|NE} setname21 [OR setname22]... ]

IF field1_SETS {EQ|NE} setname11 [OR setname12 ...]
IF field2_SETS {EQ|NE} setname21 [OR setname22 ...]
```

where:

*field1*_SETS

Is a field name with the _SETS extension that contains the names of one or more sets defined for a hierarchy.

setname11, setname12, setname21, setname22

Are the names of sets. The set names *setname11* and *setname12* are stored in *field1*_SETS. The set names *setname21* and *setname22* are stored in *field2*_SETS.

EQ

Selects members of the associated set.

NE

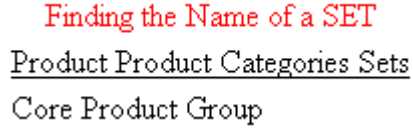
Excludes members of the associated set.

Example: Finding the Name of a Set

The following request finds the sets defined for the Products hierarchy by printing the value of its SETS field (Product_Product_Categories_SETS):

```
TABLE FILE adventure_works_mixed_case
PRINT Product_Product_Categories_SETS
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Finding the Name of a SET"
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output shows that the only set defined for this hierarchy is named *Core Product Group*:



Example: Selecting Members of a Set

The following request selects the members of the Core Product Group set from the Products hierarchy. It further limits the output with a WHERE clause on the Category1 field. ROLLUP_BY_VISUALTOTALS is not supported with this type of query, so it is set to OFF:

```
TABLE FILE adventure_works_mixed_case
SUM Internet_Sales_Amount
BY Category1
BY Subcategory1
BY Product
WHERE Product_Product_Categories_SETS EQ 'Core Product Group'
WHERE Category1 EQ 'Clothing'
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Reporting on a Set"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Reporting on a Set

Category	Subcategory	Product	Internet Sales Amount
Clothing	Caps	AWC Logo Cap	$19,688.10
	Gloves	Half-Finger Gloves, L	$10,849.07
		Half-Finger Gloves, M	$12,220.51
		Half-Finger Gloves, S	$11,951.12
	Jerseys	Long-Sleeve Logo Jersey, L	$22,595.48
		Long-Sleeve Logo Jersey, M	$22,095.58
		Long-Sleeve Logo Jersey, S	$21,445.71
		Long-Sleeve Logo Jersey, XL	$20,645.87
		Short-Sleeve Classic Jersey, L	$20,192.26
		Short-Sleeve Classic Jersey, M	$21,973.93
		Short-Sleeve Classic Jersey, S	$21,919.94
		Short-Sleeve Classic Jersey, XL	$22,081.91
	Shorts	Women's Mountain Shorts, L	$25,406.37
		Women's Mountain Shorts, M	$24,636.48
		Women's Mountain Shorts, S	$21,276.96
	Socks	Racing Socks, L	$2,427.30
		Racing Socks, M	$2,679.02
	Vests	Classic Vest, L	$12,382.50
		Classic Vest, M	$12,636.50
		Classic Vest, S	$10,668.00

Reporting on Key Performance Indicators

If a Microsoft SQL Server Analysis data source has Key Performance Indicators (KPIs) defined, you can access and display these performance measures in a WebFOCUS report.

A KPI is a calculation or set of calculations against measures in the cube that typically represent a value, status, trend, or goal. It is stored with properties that enable the organization to set and track progress toward a specified level for the KPI goal. The KPI Value represents the current state of the indicator, and the KPI Goal represents the state toward which the organization is aiming. The KPI Status and KPI Trend are derived by comparing the Goal with the Value.

If the cube has KPIs defined, any synonym created will automatically contain metadata for those KPIs.

Reference: **KPI Metadata in a Synonym**

The metadata for a KPI starts with the following KPI declaration in the Master File:

```
KPI=kpiname,
   CAPTION=kpi_caption,
   KPI_STATUS_ICON='kpi_status_graphic',
   KPI_TREND_ICON='kpi_trend_graphic'
   [,PARENT=kpi_parent_kpiname] ,$
```

where:

kpiname

Is the name of the KPI.

kpi_caption

Is the caption for the KPI.

kpi_status_graphic

Is the graphic assigned to the KPI Status property.

kpi_trend_graphic

Is the graphic assigned to the KPI Trend property.

kpi_parent_kpiname

If the KPI is part of a hierarchy of KPIs, this is the name of the parent KPI.

Field declarations describing the KPI properties follow the KPI declaration. Only the VALUE field is required, the GOAL, STATUS, and TREND fields are optional.

The following field declaration describes the KPI Value property and is required:

```
FIELDNAME=kpiname_KPI_VALUE,
   ALIAS='[Measures].[kpiname]', USAGE=Ann, ACTUAL=Ann, MISSING=ON,
   TITLE='kpi_value_title',
   REFERENCE='kpiname',
   PROPERTY=KPI_VALUE, $
```

where:

kpiname_KPI_VALUE

Is the field name for the KPI Value property.

kpiname

Is the name of the KPI.

kpi_value_title

Is the column title for the KPI Value field.

The following field declarations represent the KPI Goal, Status, and Trend properties. One or more of them may be present, but they are not required:

```
FIELDNAME=kpiname_KPI_GOAL,
   ALIAS='[Measures].[kpiname]', USAGE=Ann, ACTUAL=Ann, MISSING=ON,
   TITLE='kpi_goal_title',
   REFERENCE='kpiname',
   PROPERTY=KPI_GOAL, $

FIELDNAME=kpiname_KPI_STATUS,
   ALIAS='[Measures].[kpiname]', USAGE=Dn.m, ACTUAL=D8, MISSING=ON,
   TITLE='kpiname_status_title',
   REFERENCE='kpiname',
   PROPERTY=KPI_STATUS, $

FIELDNAME=kpiname_KPI_TREND,
   ALIAS='[Measures].[kpiname]', USAGE=Dn.m, ACTUAL=D8, MISSING=ON,
   TITLE='kpiname_trend_title',
   REFERENCE='kpiname',
   PROPERTY=KPI_TREND, $
```

where:

kpiname_KPI_GOAL, kpiname_KPI_STATUS, kpiname_KPI_TREND

Are the field names for the KPI Value, Status, and Trend properties.

kpiname

Is the name of the KPI.

kpi_goal_title, kpi_status_title, kpi_trend_title

Are the column titles for the KPI Goal, Status, and Trend fields.

Example: Sample Growth in Customer Base KPI for the Adventure Works Cube

The following declarations define the *Growth in Customer Base* KPI for the Adventure Works cube:

```
KPI=Growth in Customer Base,
   CAPTION='Growth in Customer Base',
   KPI_STATUS_ICON='Road Signs',
   KPI_TREND_ICON='Standard Arrow', $
FIELDNAME=Growth_in_Customer_Base_KPI_VALUE,
   ALIAS='[Measures].[Growth in Customer Base]', USAGE=A20, ACTUAL=A20,
MISSING=ON,
   TITLE='Growth in Customer Base Value',
   REFERENCE='Growth in Customer Base',
   PROPERTY=KPI_VALUE, $
FIELDNAME=Growth_in_Customer_Base_KPI_GOAL,
   ALIAS='[Measures].[Growth in Customer Base Goal]', USAGE=A20,
ACTUAL=A20, MISSING=ON,
   TITLE='Growth in Customer Base Goal',
   REFERENCE='Growth in Customer Base',
   PROPERTY=KPI_GOAL, $
FIELDNAME=Growth_in_Customer_Base_KPI_STATUS,
   ALIAS='[Measures].[Growth in Customer Base Status]', USAGE=D5.2,
ACTUAL=D8, MISSING=ON,
   TITLE='Growth in Customer Base Status',
   REFERENCE='Growth in Customer Base',
   PROPERTY=KPI_STATUS, $
FIELDNAME=Growth_in_Customer_Base_KPI_TREND,
   ALIAS='[Measures].[Growth in Customer Base Trend]', USAGE=D5.2,
ACTUAL=D8, MISSING=ON,
   TITLE='Growth in Customer Base Trend',
   REFERENCE='Growth in Customer Base',
   PROPERTY=KPI_TREND, $
```

Example: Sample Request Using the Growth in Customer Base KPI

The following request displays the *Growth in Customer Base* KPI value by fiscal year:

```
TABLE FILE ADVENTURE_WORKS_UPPER_CASE
WRITE GROWTH_IN_CUSTOMER_BASE_KPI_VALUE/A6
BY FISCAL_YEAR
END
```

The output is:

```
Fiscal Year Growth in Customer Base Value
----------- -----------------------------
FY 2003     0.4605
FY 2004     4.4040
FY 2005     -0.946
```

Chapter 61

Using the Adapter for Microsoft SQL Server Analysis Services Tabular Data Model

The Adapter for Microsoft® SQL Server® Analysis Services Tabular Data Model provides access to Analysis Services data sources that are deployed using the Microsoft SQL Server Analysis Services Tabular protocol, whose underlying architecture is an in-memory columnar database. The main metadata object of a Tabular database is a Model, which is a set of joined tables. A Perspective is a subset of a model.

Note that the Adapter for Microsoft SQL Server Analysis Services Tabular Data Model is for use only with Analysis Services data sources that are deployed in tabular mode. For access to Analysis Services data sources that are deployed in Multidimensional mode, use the Adapter for SQL Server Analysis Services (SSAS) available in prior releases.

The underlying language, Data Analysis Expressions (DAX), is radically different from the Multidimensional Expressions (MDX) language used with the multidimensional model.

Data within the model is not pre-aggregated, as it is with the multidimensional cube. A tabular model always contains tables and columns, while OLAP elements (such as measures, KPIs, and hierarchies) are optional. When you generate a synonym for a tabular model or perspective, the synonym describes tables as segments and columns (as well as optional measures and KPIs) as fields. If the model contains hierarchies, their structure is reflected in the synonym by properly organizing the fields that describe hierarchy levels. WebFOCUS requests against these synonyms are similar to requests against a relational data source in the sense that DAX queries generated by the adapter reference tables and columns rather than cubes and their OLAP elements (such as dimension attributes), and data selection is expressed in terms of filter predicates rather than sets of dimension members.

In this chapter:

Preparing the Microsoft SQL Server Analysis Services Tabular Data Model (TMDAX) Environment

The Adapter for SQL Server Analysis Services Tabular Data Model minimally requires the installation of the Microsoft OLE DB (MSOLAP) Provider for Analysis Services version 14.0 and up, which is part of the Microsoft Data Access Components (MDAC) installation.

No additional setup steps are required.

Procedure: How to Set Up the Environment

Microsoft OLE DB Provider for Analysis Services 14.0 must be installed in your system.

The Adapter for SQL Server Analysis Services Tabular Data Model provides read-only access to analytical data stored in in-memory columnar databases. The adapter is an OLE DB for OLAP consumer. It employs the Data Analysis Expressions (DAX) language to access tabular data.

You can use any server front-end technology with the WebFOCUS reporting engine to access the OLAP data retrieved by the server for Windows. This makes your OLAP data available to your entire enterprise. Additionally, data from Analysis Services can be joined with data from any other supported data source, providing additional information to your analytical process.

Setting SQL Server Analysis Services Tabular Data Model (TMDAX) Security

There are three methods by which you can be authenticated when connecting to SQL Server Analysis Services Tabular Data Model:

Operating System (Trusted) mode. The user ID and password used to log on to the operating system are automatically used to connect to the Analytical Engine. The server data access agent impersonates an operating system user according to the server deployment mode. The agent process establishes a connection to a SSAS tabular database based on the impersonated operating system user credentials.

SSAS Tabular Data protocol recognizes the security restrictions that the OLAP Database Administrator specifies and applies them to catalogs and cubes. This includes the role of security on the database and the application level. Although the Server for Windows establishes the connection to TMDAX, an impersonation of the client is used to maintain access privileges.

- **Security off.** If the Server is running with security off, the user ID and password of user that started the Reporting Server is passed to the Analytical Engine in order to establish a connection and access rights. This is the logon ID specified for the Windows or Windows Workstation after it is first started.

- **Security on.** If the Server is running with security on, the system credentials of the client connecting to the server is passed to the Analytical Engine in order to establish a connection and access privileges. This process is called impersonation.

Explicit mode. The user ID and password are explicitly specified for each connection and passed to SQL Server Analysis Services, at connection time, for authentication

Password passthru mode. The user ID and password are explicitly specified for each connection and passed to SQL Server Analysis Services, at connection time, for authentication.

Accessing SQL Server Analysis Services

Using the standard rules for deploying OLE DB, the server supports connections to:

- Local SQL Server Analysis Services.
- Remote SQL Server Analysis Services. (To connect to a remote TMDAX instance, you must have Microsoft OLE DB Provider for Analysis Services 15.0 installed in your system.)

Configuring the Adapter for Microsoft SQL Server Analysis Services Tabular Data Model

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Microsoft SQL Server Analysis Services Tabular database server, the adapter requires connection and authentication information.

You enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the SET CONNECTION_ATTRIBUTES command to the profile you select, which can be the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).

You can create connections to more than one Microsoft SQL Server Analysis Services Tabular database by configuring multiple connections. The actual connection to the Microsoft SQL Server Analysis Services Tabular database takes place when the first query that references the connection is issued. If you configure multiple connections:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Microsoft SQL Server Analysis Services Tabular Data Model

The *SSAS Tabular Data Model* adapter is under the *OLAP* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server

Name of the machine where Microsoft SQL Server Analysis Services is running. If that machine has more than one instance of SQL Server Analysis Services installed, provide the server name and instance name as follows: server\instance

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication as a standard login.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication as a standard login.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Default Database

Name of the Microsoft SQL Server Analysis Services Tabular database used for this connection. The database name, including path, must be enclosed in single quotation marks. This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.

Additional connection string keywords (optional)

Please refer to Microsoft SQL Server Analysis Services documentation for all available connection string keywords used to change the behavior of the TMDAX connection.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Managing Microsoft SQL Server Analysis Services Tabular Data Model Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server Analysis Services data types.

Creating Synonyms

Reference: Synonym Creation Parameters for Microsoft SQL Server Analysis Services Tabular Data Model

The following list describes the synonym creation parameters for which you can supply values.

Select the Database

Provides a drop-down list of models and perspectives available for synonym creation. The default selection is *All Databases*.

Filter by Object (Model or Perspective) Name

Selecting this option opens a text box in which you can enter a name or a pattern. Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Click *Next* to open the *Available Objects for SSAS Tabular Model* page. The following parameters are available:

Application

Select an application directory. The default value is ibisamp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Database

Is the name of the database.

Object Name

Is the name of the underlying object.

Type

Is the type of object, either MODEL or PERSPECTIVE.

Select Objects

Select objects for which you wish to create synonyms:

- To select all objects in the list, select the *Select All* check box.
- To select specific objects, select the corresponding check boxes.

Click *Create Base Synonyms* on the ribbon.

Example: Sample Generated Synonym

An Adapter for Microsoft SQL Server Analysis Services Tabular Data Model synonym comprises a Master File and an Access File. This is a synonym for the perspective named WF Retail Tabular Cube.

Generated Master File wf_retail_tabular_cube.mas

```
FILENAME=, SUFFIX=TMDAX   , $
  SEGMENT=WRD_WF_RETAIL_SALES, SEGTYPE=S0, $
    FIELDNAME=SUM_OF_COGS_US, ALIAS='Sum of COGS_US', USAGE=D18.2C,
ACTUAL=D8,
      TITLE='Sum of COGS_US',
      PROPERTY=MEASURE,  $
    FIELDNAME=ID_SALES, ALIAS=ID_SALES, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_SALES', $
    FIELDNAME=ID_STORE, ALIAS=ID_STORE, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_STORE', $
    FIELDNAME=ID_CURRENCY, ALIAS=ID_CURRENCY, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_CURRENCY', $
    FIELDNAME=ID_CUSTOMER, ALIAS=ID_CUSTOMER, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_CUSTOMER', $
    FIELDNAME=ID_DISCOUNT, ALIAS=ID_DISCOUNT, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_DISCOUNT', $
    FIELDNAME=ID_PRODUCT, ALIAS=ID_PRODUCT, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_PRODUCT', $
    FIELDNAME=ID_TIME, ALIAS=ID_TIME, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_TIME', $
    FIELDNAME=COGS_LOCAL, ALIAS=COGS_LOCAL, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='COGS_LOCAL', $
```

```
FIELDNAME=COGS_US, ALIAS=COGS_US, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='COGS_US', $
    FIELDNAME=DISCOUNT_LOCAL, ALIAS=DISCOUNT_LOCAL, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='DISCOUNT_LOCAL', $
    FIELDNAME=DISCOUNT_US, ALIAS=DISCOUNT_US, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='DISCOUNT_US', $
    FIELDNAME=GROSS_PROFIT_LOCAL, ALIAS=GROSS_PROFIT_LOCAL, USAGE=D20.2,
ACTUAL=D8,
      MISSING=ON,
      TITLE='GROSS_PROFIT_LOCAL', $
    FIELDNAME=GROSS_PROFIT_US, ALIAS=GROSS_PROFIT_US, USAGE=D20.2,
ACTUAL=D8,
      MISSING=ON,
      TITLE='GROSS_PROFIT_US', $
    FIELDNAME=MSRP_LOCAL, ALIAS=MSRP_LOCAL, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='MSRP_LOCAL', $
    FIELDNAME=MSRP_US, ALIAS=MSRP_US, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='MSRP_US', $
    FIELDNAME=QUANTITY_SOLD, ALIAS=QUANTITY_SOLD, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='QUANTITY_SOLD', $
    FIELDNAME=REVENUE_LOCAL, ALIAS=REVENUE_LOCAL, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='REVENUE_LOCAL', $
    FIELDNAME=REVENUE_US, ALIAS=REVENUE_US, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='REVENUE_US', $
```

```
  SEGMENT=WRD_WF_RETAIL_CUSTOMER, SEGTYPE=U, PARENT=WRD_WF_RETAIL_SALES,
    JOIN_WHERE=WRD_WF_RETAIL_SALES.ID_CUSTOMER EQ
WRD_WF_RETAIL_CUSTOMER.ID_CUSTOMER;, $
    FIELDNAME=ID_CUSTOMER, ALIAS=ID_CUSTOMER, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_CUSTOMER', $
    FIELDNAME=ID_AGE, ALIAS=ID_AGE, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_AGE', $
    FIELDNAME=ID_EDUCATION, ALIAS=ID_EDUCATION, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_EDUCATION', $
    FIELDNAME=ID_GEOGRAPHY, ALIAS=ID_GEOGRAPHY, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_GEOGRAPHY', $
    FIELDNAME=ID_INCOME, ALIAS=ID_INCOME, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_INCOME', $
    FIELDNAME=ID_INDUSTRY, ALIAS=ID_INDUSTRY, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_INDUSTRY', $
    FIELDNAME=ID_MARITAL_STATUS, ALIAS=ID_MARITAL_STATUS, USAGE=P20,
ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_MARITAL_STATUS', $
    FIELDNAME=ID_OCCUPATION, ALIAS=ID_OCCUPATION, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_OCCUPATION', $
    FIELDNAME=ID_TIME_MIN, ALIAS=ID_TIME_MIN, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_TIME_MIN', $
    FIELDNAME=ID_TIME_MAX, ALIAS=ID_TIME_MAX, USAGE=P20, ACTUAL=P10,
      MISSING=ON,
      TITLE='ID_TIME_MAX', $
    FIELDNAME=EMAIL_ADDRESS, ALIAS=EMAIL_ADDRESS, USAGE=STRING,
ACTUAL=STRING,
      MISSING=ON,
      TITLE='EMAIL_ADDRESS', $
    FIELDNAME=FIRSTNAME, ALIAS=FIRSTNAME, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='FIRSTNAME', $
    FIELDNAME=FULLNAME, ALIAS=FULLNAME, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='FULLNAME',
      WITHIN=POSTAL_CODE, $
```

```
    FIELDNAME=GENDER, ALIAS=GENDER, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='GENDER', $
    FIELDNAME=LASTNAME, ALIAS=LASTNAME, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='LASTNAME', $
    FIELDNAME=INCOME, ALIAS=INCOME, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON,
      TITLE='INCOME', $
    FIELDNAME=POSTAL_CODE, ALIAS=POSTAL_CODE, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='POSTAL_CODE',
      WITHIN=CITY,
      GEOGRAPHIC_ROLE=POSTAL-CODE,  $
    FIELDNAME=CITY, ALIAS=CITY, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='CITY',
      WITHIN=STATE_PROVINCE,
      GEOGRAPHIC_ROLE=CITY,  $
    FIELDNAME=STATE_PROVINCE, ALIAS=STATE_PROVINCE, USAGE=STRING,
ACTUAL=STRING,
      MISSING=ON,
      TITLE='STATE_PROVINCE',
      WITHIN=COUNTRY,
      GEOGRAPHIC_ROLE=STATE,  $
    FIELDNAME=COUNTRY, ALIAS=COUNTRY, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='COUNTRY',
      WITHIN=CONTINENT,
      GEOGRAPHIC_ROLE=COUNTRY,  $
    FIELDNAME=CONTINENT, ALIAS=CONTINENT, USAGE=STRING, ACTUAL=STRING,
      MISSING=ON,
      TITLE='CONTINENT',
      WITHIN='*CUSTOMERS',
      GEOGRAPHIC_ROLE=CONTINENT,  $
 DIMENSION=CUSTOMERS, $
  HIERARCHY=CUSTOMERS, HRY_DIMENSION=CUSTOMERS, $
```

Generated Access File wf_retail_tabular_cube.acx

```
MODELNAME=WF Retail Tabular Cube,
  CONNECTION=CON01,
  CATALOG=WF Retail, $
 SEGNAME=WRD_WF_RETAIL_SALES,
   TABLENAME='''WRD_WF_RETAIL_SALES''', $
  FIELD=SUM_OF_COGS_US,
    MEASURE=YES, $
 SEGNAME=WRD_WF_RETAIL_CUSTOMER,
   TABLENAME='''WRD_WF_RETAIL_CUSTOMER''', $
```

The following request reports against this synonym.

```
TABLE FILE wf_retail_tabular_cube
SUM COGS_US REVENUE_US
BY CONTINENT
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

The output is shown in the following image.

CONTINENT	COGS_US	REVENUE_US
Africa	2,368,167.00	3,296,291.87
Asia	27,894,318.00	38,804,729.21
Europe	373,105,084.00	520,215,112.82
North America	318,808,445.00	444,147,282.83
Oceania	1,837,474.00	2,562,553.72
South America	37,426,041.00	52,166,954.75

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Chapter 62

Using the Adapter for Microsoft SQL Server ODBC

The Adapter for Microsoft® SQL Server® ODBC allows applications to access Microsoft SQL Server data sources. The adapter converts data or application requests into native Microsoft SQL Server statements and returns optimized answer sets to the requesting program.

In this chapter:

Preparing the Microsoft SQL Server ODBC Environment

The Microsoft SQL Server environment is set up during the installation of the Microsoft SQL Server 2012 or 2014 and the ODBC Driver 11 or higher for SQL Server.

No additional setup steps are required.

Accessing Microsoft SQL Server Remotely

You can access Microsoft SQL Server on a remote node. To access Microsoft SQL Server remotely, you must:

- Locally install the latest version of Microsoft SQL Server Native Client and ODBC Driver 11 or higher for SQL Server.
- Know the name of the remote Microsoft SQL Server instance.

All Microsoft SQL Servers installed are defined by using a unique NetBEUI name. The server can access any Microsoft SQL Server on the network, provided you define a valid user ID and password, as well as the name of the Microsoft SQL Server instance. You can define these parameters in either the server global profile or a user profile.

Configuring the Adapter for Microsoft SQL Server ODBC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Microsoft SQL Server database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command.

You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Microsoft SQL Server database by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Microsoft SQL Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Microsoft SQL Server ODBC

The *MS SQL Server ODBC* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server

Name of the machine where Microsoft SQL Server is running. If that machine has more than one instance of SQL Server installed, provide the server name and instance name as follows: server\instance

Security

There are three methods by which a user can be authenticated when connecting to Microsoft SQL Server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication as a standard login. This option requires that SQL Server security be set to SQL Server and Windows.

- **Password Passthru.** The user ID and password received from the client application are passed to Microsoft SQL Server, at connection time, for authentication as a standard login. This option requires that SQL Server security be set to SQL Server and Windows.
- **Trusted.** The adapter connects to Microsoft SQL Server as an operating system login using the credentials of the operating system user impersonated by the server data access agent. This option works with either of SQL Server security settings

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Default Database

Name of the Microsoft SQL Server database used for this connection. The database name, including path, must be enclosed in single quotation marks. This parameter is optional. If not specified, it defaults to the database associated with the authorization ID.

Additional connection string keywords (optional)

Please refer to Microsoft SQL Server documentation for all available connection string keywords used to change the behavior of the ODBC connection.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually on Windows

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication.

```
ENGINE MSODBC SET CONNECTION_ATTRIBUTES [connection]
 server/userid,password [;dbname]
[;additional_connection_string_keywords]
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication.

```
ENGINE MSODBC SET CONNECTION_ATTRIBUTES [connection]
 server/[;dbname][;additional_connection_string_keywords]
```

Trusted authentication. The adapter connects to Microsoft SQL Server as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

```
ENGINE MSODBC SET CONNECTION_ATTRIBUTES [connection]server/,[;dbname]
[;additional_connection_string_keywords]
```

Note: Enclose values that contain special characters in single quotation marks. If a value contains a single quotation mark, this quotation mark must be preceded by another single quotation mark, resulting in two single quotation marks in succession. For example, to specify the user ID Mary O'Brien, which contains both a blank and a single quotation mark, enter: 'Mary O' 'Brien'.

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE MSODBC SET DEFAULT_CONNECTION connection
```

where:

`MSODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.

- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE MSODBC SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE MSODBC SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`MSODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Microsoft SQL Server ODBC Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server ODBC data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Microsoft SQL Server ODBC table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

The adapter supports the creation of metadata for certain types of MS SQL Native Synonyms. This feature is only available for Microsoft SQL Server versions 2012 or higher.

You can create metadata for Native Synonyms under the following conditions:

- Native Synonyms must be created from a TABLE or VIEW base object.
- The base object (TABLE or VIEW) must exist on the MS SQL Server targeted by the adapter connection string.
- Native Synonyms must be created with the base object described by a name consisting of not more than three components. Depending on the location and ownership of the base object within the targeted server, acceptable formats, are:

  ```
  object_name, schema_name.object_name,
  ```

  or

  ```
  database_name.schema_name.object_name.
  ```

The following types of Native Synonyms are not supported:

- Those based on stored procedures.
- Those based on objects existing on a linked MS SQL server.
- Those based on objects described with a 4-part name, such as:

  ```
  server_name.database_name.schema_name.object_name
  ```

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 1541.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Microsoft SQL Server ODBC

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Important: If you select Stored Procedures as your object type, the input parameters will be a little different from those described here. For details, refer to *Creating a Report Against a Stored Procedure* on page 1544.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check Use current database to use the database that has been set as the default database.
- Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

  Before selecting a database, if Use current database is checked, uncheck it.

To specify the intended database, choose from the Select database drop-down menu, which shows all databases on the targeted instance of Microsoft SQL Server. Selecting Default Database will retain the database set during connection configuration. If Default Database was not set during configuration, the database assigned to the active login on the SQL Server will be used as the default.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Microsoft SQL Server ODBC Data Type Support* on page 1539.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Microsoft SQL Server ODBC synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=MSODBC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=edaqa.nf29004,
CONNECTION=connmss, KEYS=1, WRITE=YES,$
```

Reference: Mapping Microsoft SQL ODBC Table Comments Into a Synonym

When you generate a synonym for a Microsoft SQL table or view, the adapter maps comments as follows:

- MS SQL Server table/view comments (if present) are mapped to the REMARKS attribute in the Master File synonym.
- MS SQL Server column comments (if present) are mapped to the DESCRIPTION attribute in the Master File synonym.

Both Unicode and non-Unicode comments are supported.

Also, MS SQL Server column title (if present) is mapped to the TITLE attribute in the Master File synonym.

Reference: **Access File Keywords**

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Microsoft SQL Server table. The table name can be fully qualified as follows: TABLENAME=[[*database*.]*owner*.]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them first in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	When set to *Yes*, enables the auto increment feature.
START	Initial value in incrementing sequence.
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Always Encrypted Support**

The Windows ODBC version of the Adapter for SQL Server supports the SQL Server 2017/2016 Always Encrypted native feature under the following conditions:

❑ The columns have to be encrypted using SQL Server Management Studio prior to creating any synonyms.

- The adapter connection is configured with the connection string keyword ColumnEncryption=Enabled.

When a synonym is created, the Access File will contain the attribute ENCRYPT_TYPE={DETERMINISTIC|RANDOMIZED} for the encrypted columns.

Certain operations on encrypted columns cannot be performed in SQL. The adapter logic will account for that, and those operations will be performed by the WebFOCUS Reporting Server.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Microsoft SQL Server ODBC Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE MSODBC SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Support of Read-Only Fields

CREATE SYNONYM creates a field description with FIELDTYPE=R for Microsoft SQL Server columns created as TIMESTAMP or columns with the IDENTITY attribute. These fields are read-only. When executing a MAINTAIN or MODIFY procedure, the adapter suppresses all write operations against columns marked in the Master File with FIELDTYPE=R.

Example: **Supporting a Read-Only Field**

This example creates a table in which the first column has the IDENTITY property and the second column is a timestamp column:

```
CREATE TABLE TAB1
 (idproptab int IDENTITY (1,1), timstmp timestamp)
```

CREATE SYNONYM generates the following Master File for this table:

```
FILE=TAB1, SUFFIX=MSODBC ,$
SEGNAME=TAB1, SEGTYPE=S0 ,$
FIELD=IDPROPTAB, idproptab, I11, I4,  MISSING=OFF, FIELDTYPE=R ,$
FIELD=TIMSTMP,   timstmp,   A16, A16, MISSING=ON,  FIELDTYPE=R ,$
```

Reporting Against a Microsoft SQL Server ODBC Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Microsoft SQL Server ODBC stored procedures and report against the procedure output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters: OUT parameters, and INOUT parameters in OUT mode, as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 1541.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 1544.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

A synonym describes the stored procedure parameters and answer set.

An answer set structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, *input parameters* refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception. If you know the internal logic of the procedure, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

Example: Synonym for Microsoft SQL Server ODBC Stored Procedure CustOrders

The following synonym describes a Microsoft SQL Server ODBC stored procedure with one input parameter, one output parameter, and one answer set containing four variables.

The Master File synonym is:

```
FILENAME=CUSTORDERS, SUFFIX=MSODBC  , $
   SEGMENT=INPUT, SEGTYPE=S0, $
      FIELDNAME=@CUSTOMERID, ALIAS=P0001, USAGE=A5, ACTUAL=A5,
         MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
   SEGMENT=OUTPUT, SEGTYPE=S0, PARENT=INPUT, $
      FIELDNAME=@RETURN_VALUE, ALIAS=P0000, USAGE=I11, ACTUAL=I4, $
   SEGMENT=ANSWERSET1, SEGTYPE=S0, PARENT=INPUT, $
      FIELDNAME=ORDERID, ALIAS=OrderID, USAGE=I11, ACTUAL=I4, $
      FIELDNAME=ORDERDATE, ALIAS=OrderDate, USAGE=HYYMDs, ACTUAL=HYYMDs,
         MISSING=ON, $
      FIELDNAME=REQUIREDDATE, ALIAS=RequiredDate, USAGE=HYYMDs,
         ACTUAL=HYYMDs, MISSING=ON, $
      FIELDNAME=SHIPPEDDATE, ALIAS=ShippedDate, USAGE=HYYMDs,
         ACTUAL=HYYMDs, MISSING=ON, $
```

The Access File synonym is:

```
SEGNAME=INPUT, CONNECTION=ITarget, STPNAME=Northwind.dbo.CustOrders, $
SEGNAME=OUTPUT, STPRESORDER=0, $
SEGNAME=ANSWERSET1, STPRESORDER=1, $
```

Reference: Synonym Creation Parameters for Stored Procedures

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Select *Stored Procedures*.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check Use current database to use the database that has been set as the default database.
- Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

Before selecting a database, if Use current database is checked, uncheck it.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Note: For Db2, this applies to all platforms except IBM i.

Select

Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.

Name

The name of the synonym, which defaults to the stored procedure name.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters.

If all procedures have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Microsoft SQL Server ODBC Data Type Support* on page 1539.

Values

Select the check box for *every* parameter displayed for the specified procedure.

Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines:

- Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided.
- Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym creation purposes.

If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.

Creating a Report Against a Stored Procedure

You can report against a stored procedure answer set using the same facilities you use to report against a database table:

- **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 1546.
- **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 1544.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

IF

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

WHERE

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Microsoft SQL Server ODBC Environment

The Adapter for Microsoft SQL Server ODBC provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Microsoft SQL Server.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE MSODBC SET TIMEOUT {nn|0}
```

where:

`MSODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Specifying the Login Wait Time

You can use the LOGINTIMEOUT command to specify the number of seconds the adapter will wait for a response from Microsoft SQL Server at connect time.

Note: For compatibility with previous releases of the adapter, TIMEOUT is available as a synonym for LOGINTIMEOUT.

Syntax: How to Specify the Login Wait Time

```
ENGINE MSODBC SET LOGINTIMEOUT|TIMEOUT {nn|0}
```

where:

MSODBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. The default value is approximately 15 seconds.

0

Represents an infinite period to wait for login response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

The available parameters are:

```
ENGINE MSODBC SET PASSRECS {ON|OFF}
```

where:

MSODBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE MSODBC SET ISOLATION {RU|RC|RR|SE|CH|CS}
```

where:

`RU`

Sets the transaction isolation level to Read Uncommitted.

`RC`

Sets the transaction isolation level to Read Committed.

`RR`

Sets the transaction isolation level to Repeatable Read.

`SE`

Sets the transaction isolation level to Serializable Read.

`CH`

Sets the transaction isolation level to Chaos.

`CS`

Sets the transaction isolation level to Cursor Stability, which is a synonym for Read Committed.

Microsoft SQL Server ODBC Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Optimizing Requests if a Virtual Field Contains Null Values

The SET OPTNOAGGR command provides finely-tuned control of adapter behavior for optimization. Users who for any reason wish to prevent passing aggregation to the RDBMS can use this command. An example of such a reason might be where NULL values occur in aggregated data with calculations. The SET OPTNOAGGR command causes the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is instead performed internally by the server while JOIN and SORT operations are handled by the RDBMS.

If any DEFINE field contains calculations with NULL fields then such operations cannot be translated to SQL and pass to DBMS because always return NULL. It has to be processed by FOCUS.

This can be achieved by SET OPTIMIZATION OFF.

However, in some cases it is preferable to use the off-load JOIN and SORT operation to DBMS for better performance while leaving AGGREGATION to FOCUS.

Syntax: How to Set Enhanced Aggregation Control

```
SQL MSODBC SET OPT {AGGR|NOAGGR}
```

where:

AGGR

Directs the adapter to off-load aggregated DEFINE fields to the DBMS. This is the default setting.

NOAGGR

Directs the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is, instead, performed internally by the server, while JOIN and SORT operations are handled by the RDBMS. This setting can also be used to provide backwards compatibility for applications that were written based on the functionality of the previous release, when less SQL was off-loaded to the RDBMS. For example, when a calculation on aggregated fields may have contained NULL data that was not processed by the RDBMS NVL( ) function.

Example: **Using IF-THEN_ELSE Optimization With a Condition That Is Always False**

```
SQL MSODBC SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR FETCH ONLY;
```

Reference: **SQL Limitations on Optimization of DEFINE Expressions**

Since the FOCUS reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- User-written subroutines.
- Self-referential expressions, such as:

  ```
  X=X+1;
  ```

- EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- DECODE functions for field value conversions.
- Relational operators INCLUDES and EXCLUDES.
- FOCUS subroutines ABS, INT, MAX, MIN, LOG, and SQRT.

  Note: Do not confuse the FOCUS user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.
- Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- Date-time manipulation handled by the FOCUS date-time functions is not converted to SQL.
- Financial Modeling Language (FML) cell calculations. FML is also referred to as Extended Matrix Reporting (EMR).

Note: FML report requests are extended TABLE requests. The Financial Modeling Language provides special functions for detailed reporting. Consult your FOCUS documentation for more information.

In addition, IF-THEN-ELSE optimization does not support the following features:

- Any type of DECODE expression.
- STATIC SQL.
- IF/WHERE DDNAME.
- Partial date selection.

Specifying Block Size for Retrieval Processing

The Adapter for Microsoft SQL Server ODBC supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE MSODBC SET FETCHSIZE n
```

where:

`MSODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE MSODBC SET INSERTSIZE n
```

where:

MSODBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE MSODBC SET FASTLOAD [ON|OFF]
```

where:

MSODBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Uses the Bulk Insert API. ON is the default.

OFF

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Microsoft SQL Server ODBC.

For the Adapter for Microsoft SQL Server ODBC, the Bulk API is used automatically in LOADONLY mode. Measurements show that intermediate flushes do not affect performance; therefore, the behavior does not depend on the INSERTSIZE.

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Optimizing Non-Equality WHERE-Based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

`LEFT_OUTER`

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

`hostfile`

Is the host Master File.

`AT`

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

`hfld1`

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

tag1

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

WITH *hfld2*

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

MULTIPLE

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

UNIQUE

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations. It always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: Conditions for WHERE-Based Outer Join Optimization

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left Outer Join

The following request creates a left outer conditional join between two MSSQL data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE employee AT EMPLOYEEID
TO ALL FILE employeepayhistory AT EMPLOYEEID
     WHERE RATECHANGEDATE GT HIREDATE;
END
TABLE FILE employee
PRINT RATE
BY EMPLOYEEID
END
```

The WebFOCUS request is translated to a single MSSQL SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."EmployeeID", T1."HireDate", T2."EmployeeID",
T2."RateChangeDate", T2."Rate" FROM
AdventureWorks.HumanResources.Employee T1 LEFT OUTER JOIN
AdventureWorks.HumanResources.EmployeePayHistory T2 ON
(T2."RateChangeDate" > T1."HireDate") ) ORDER BY T1."EmployeeID";
```

Improving Optimizer Efficiency with Hints

DBMS Optimizer hints can be used to alter an execution plan. The adapter provides a setting which enable the TABLE command to place the hints at the end of the generated query for Microsoft SQL Server.

This occurs when the adapter constructs a single SELECT statement. It does not occur in the case of a FOCUS-managed Join when multiple SELECTs are generated.

To reverse the setting, use SET HINT without a hint_text parameter.

Syntax: How to Set Specific Hints

Use the following syntax to set specific hints:

```
SQL MSODBC SET HINT OPTION (hint_text)
```

where

`MSODBC`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`OPTION (hint_text)`

Is the text of the hint or hints combination. The end user is responsible for the syntax. Omitting OPTION (*hint_text*) resets the hint to none.

Example: Setting an Microsoft SQL Hint

```
SQL MSODBC SET HINT OPTION (FAST 2)
TABLE FILE STXT31M
 PRINT *
 BY F01INT
 END
```

The WebFOCUS request is translated into a SELECT statement that incorporates the specified hint.

```
SELECT
   T1."F01INT",
   T1."F02CHAR_10",
   T1."F03VARCHAR_10"
FROM
   D999AIXPPC71XX_TTXT31M T1
ORDER BY
   T1."F01INT"
OPTION (FAST 2);
```

Calling a Microsoft SQL Server ODBC Stored Procedure Using SQL Passthru

Microsoft SQL Server stored procedures are supported using SQL Passthru. These procedures need to be developed within Microsoft SQL Server using the CREATE PROCEDURE command.

The adapter supports stored procedures with input, output, and in-out parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets are returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to the rules of the underlying DBMS. Note that the examples shown in this section are SQL-based. See the DBMS documentation for rules, languages, and additional programming examples.

Syntax: How to Invoke a Stored Procedure

```
SQL MSODBC EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

`MSODBC`

Is the ENGINE suffix for Microsoft SQL Server.

`procname`

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

You can employ either SQL or SYS naming conventions to control the separator character used for interpreting multipart names, as described in Setting Naming Conventions.

parameter_specification

IN, OUT, and INOUT parameters are supported. Use the variation required by the stored procedure:

IN

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

OUT

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

INOUT

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

In this example, a user invokes a stored procedure, edaqa.test_proc01, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL MSODBC EX edaqa.test_proc01 125,?,?/3.14,,'abc',,'xyz'
END
```

Example: Sample Stored Procedure

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (   OUT chSQLSTATE_OUT   CHAR(5),
                                  OUT intSQLCODE_OUT   INT,
                                  INOUT l_name char(20),
                                  INOUT f_name char(20))
    RESULT SETS 1
    LANGUAGE SQL
-------------------------------------------------------------------------
-- SQL Stored Procedure
-------------------------------------------------------------------------
P1: BEGIN
    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT
           EDAQA.NF29005.SSN5 AS SSN5,
           EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
           EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
           EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
           EDAQA.NF29005.SEX5 AS SEX5
        FROM
           EDAQA.NF29005
        WHERE
           (
             ( EDAQA.NF29005.LAST_NAME5 =  l_name )
        AND
             ( EDAQA.NF29005.FIRST_NAME5 = f_name )
           );
     -- Cursor left open for client application
    OPEN cursor1;
    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';
END P1   @
```

Reference: **Capturing Application Errors in Stored Procedures**

You can capture application errors using the RAISERROR method. Any application error that is issued by the stored procedure is available in the server variable &MSOMSGTXT.

Chapter 63

Using the Adapter for Millennium

The Adapter for Millennium allows applications to access Millennium data sources. The adapter converts application requests into native Millennium statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Server Environment for Millennium

The Adapter for Millennium supports compressed Millennium VSAM files. The Millennium decompression routines have been added through the existing Zcomp1 exit point. To enable support for compressed files, include the following set command is in your server profile, EDASPROF.prf:

```
ENGINE CPMILL SET ZCOMP FOCMILZ
```

The adapter requires two control files to be built and allocated to DDNAME CPMILL and CPMILLI via JCL or DYNAM allocation. The Millennium control file is used as input to these files and is dynamically allocated to the server only for the creation of CPMILL and CPMILLI.

Procedure: How to Prepare the Server Environment for Millennium

Allocate two MVS PDSs, CPMILL and CPMILLI, with no DCB information. DCB information is determined by the server at run time. The space requirements are:

- First extent cylinders: 5
- Secondary cylinders: 2

Sample JCL for CPMILL allocation follows:

```
//ALOCLIB EXEC PGM=IEFBR14
//APPL  DD DSN=HIGHLQ.CPMILL,DISP=(NEW,CATLG),
//      VOL=SER=USERMC,UNIT=3390,SPACE=(CYL,(5,2))
```

Sample JCL for CPMILLI allocation follows:

```
//ALOCLIB EXEC PGM=IEFBR14
//APPL  DD DSN=HIGHLQ.CPMILLI,DISP=(NEW,CATLG),
//      VOL=SER=USERMC,UNIT=3390,SPACE=(CYL,(5,2))
```

Configuring the Adapter for Millennium

You can configure the adapter from the Web Console or the Data Management Console. You must then edit and execute several files to complete the configuration.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Procedure: How to Edit Configuration Files for the Adapter for Millennium

After configuring the adapter from the Web Console, perform the following additional steps:

1. Edit the server profile, EDASPROF, to:

   - ❑ Allocate CPMILL and CPMILLI to the server.
   - ❑ Add your GEAC VSAM files, as in the following sample.

```
-************************************
  ENGINE CPMILL SET ZCOMP FOCMILZ
-************* GEAC Allocations*********
   DYNAM ALLOC FILE CPMILL  MOD -
    DATASET HIGHLQ.CPMILL
   DYNAM ALLOC FILE CPMILLI MOD -
    DATASET HIGHLQ.CPMILLI
   DYNAM ALLOC FILE HRMH0B  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33EM1
   DYNAM ALLOC FILE H33EM2  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33EM2
   DYNAM ALLOC FILE H33EM3  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33EM3
   DYNAM ALLOC FILE HRMH7O  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33PER
   DYNAM ALLOC FILE HRMH7S   SHR REU -
    DATASET MKTPJC.GEAC.HR.H33PER
   DYNAM ALLOC FILE H33QEH1  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33QEH1
   DYNAM ALLOC FILE H33QEH2  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33QEH2
   DYNAM ALLOC FILE H33QEH3  SHR REU -
    DATASET MKTPJC.GEAC.HR.H33QEH3
   DYNAM ALLOC FILE H33TAX   SHR REU -
    DATASET MKTPJC.GEAC.HR.H33TAX
   DYNAM ALLOC FILE H33UTL   SHR REU -
```

2. Start the server.
3. Using RDAAPP on MVS or the Test Tool on NT, execute the CPMILLI RPC, located in the IBIFEX DDNAME allocation of the server. This procedure uses the Millennium control file as input to populate the CPMILLI control file.

   To execute the RPC, you need the dataset name of the Millennium control file (for example, M3000):

```
CPMILLI DSN=GEAC.MILL.M30000
```

   After execution, CPMILLI will contain the following records:

```
CDBAMDM3LL_______
CDBAMIM3LL_______
CDBAMNM3LL_______
CDBH0BM3LL_______
CDBH7OM3LL_______
CDBH7SM3LL_______
```

4. Edit CPMILLI to associate the Millennium DB_DBIDs, DB_DBSUBIDs, and DB_TRANIDs with the names of the corresponding Master Files, which are delivered with the Millennium product.

CPMILLI Record	Master File Name	Edited Result
CDBAMDM3LL	APMAMD	CDBAMDM3LLAPMAMD
CDBAMIM3LL	APMAMI	CDBAMIM3LLAPMAMI
CDBAMNM3LL	APMAMN	CDBAMNM3LLAPMAMN
CDBH0BM3LL	HRMH0B	CDBH0BM3LLHRMH0B
CDBH7OM3LL	HRMH0B	CDBH7OM3LLHRMH7O
CDBH7SM3LL	HRMH0B	CDBH7SM3LLHRMH7S

5. Using RDAAP on MVS or the Test Tool on NT, execute the CPMILL RPC, located in the IBIFEX DDNAME allocation of the server. This procedure associates the Millennium control file information with the adapter routines.

   CPMILL takes two input parameters: &dsn for the CPMILLI dataset, and &dsn1 for the Millennium control file:

   ```
   CPMILL DSN=HIGHLQ.CPMILLI,DSN1=GEAC.MILL.M30000
   ```

6. Edit the server profile, EDASPROF, to change the file allocation for CPMILL and CPMILLI from MOD to SHR REU, as shown below:

   ```
   -*************************************
     ENGINE CPMILL SET ZCOMP FOCMILZ
   -************* GEAC Allocations*********
      DYNAM ALLOC FILE CPMILL  SHR REU -
       DATASET HIGHLQ.CPMILL
      DYNAM ALLOC FILE CPMILLI SHR REU -
       DATASET HIGHLQ.CPMILLI
   ```

Preparing the Millennium Environment

Prior to using the server and configuring the Adapter for Millennium using the Web Console, you need to edit the server IRUNJCL procedure.

Procedure: How to Edit the Server IRUNJCL

Procedure overview:

1. Concatenate the Millennium API modules for the adapter to DDNAME STEPLIB.
2. Allocate the DDNAMEs M30000 and M30002 to your Millennium Control File data sets, if you are running Millennium Release 3.
3. Determine the DDNAMEs for the Millennium databases based on your Millennium installation options. (See your Millennium documentation or contact your Millennium administrator for this information.)
4. Allocate DDNAME FOCPRV to the data set that contains Access Files. You can omit the allocation for DDNAME FOCPRV if all database IDs are assigned using the Master File member name (for more information, see *Managing Millennium Metadata* on page 1566), or access the default lead transaction ID.

   Your system support staff can supply proper data set names for your site. The Millennium API modules must be MVS-loadable.

The server must allocate the specific files of the adapter in the IRUNJCL, as well as in a global profile (EDASPROF.CFG).

Edit the IRUNJCL and EDASPROF.CFG as follows:

IRUNJCL. Millennium load library and Control files and databases:

```
//M30000 DD DISP=SHR,DSN=millen.ctrlfile.M30000
//M30002 DD DISP=SHR,DSN=millen.ctrlfile.M30002
//database DD DISP=SHR,DSN=millen.database
```

The IRUNJCL must have the following data sets allocated to the following DDNAMEs.

EDASPROF.CFG. Millennium Access File data set allocation

```
//* Millennium ALLOCATIONS
DYNAM ALLOC F FOCPRV DA qualif.FOCPRV.DATA SHR REUSE
```

where:

millen

Is the high-level qualifier for your Millennium production data sets.

ctrlfile

Is the Millennium control file.

database

Is the DDNAME required by Millennium for a Millennium database. If you access more than one Millennium database, you must allocate more than one DDNAME, each determined by the options chosen when your site installed Millennium.

qualif

Is the high-level qualifier for your server production data sets.

Adapter Tracing

To activate the Adapter for Millennium Tracing Facility, you must allocate DDNAME GPTRACE in the server JCL:

```
//GPTRACE DD SYSOUT=*
```

Note: Use the adapter Tracing Facility carefully. The DDNAME is allocated to the server JCL; therefore, it is applied globally. This causes the adapter activity to be traced for all server tasks. This facility should be used under tightly controlled testing circumstances.

Managing Millennium Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Millennium data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Millennium file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Millennium

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

COBOL FD Selection Options

File System

Select one of the following from the drop-down list:

- *Fully qualified PDS name* to enter a fully qualified MVS Library in the entry box.
- *Absolute HFS directory pathname* to enter the HFS location that contains the COBOL FD.

PDS name or Directory name

Depending on your selection for File System, enter the fully qualified PDS name or absolute directory path that contains the COBOL FD.

Member Name or File Name

Depending on your selection for File System, enter the member name or file that contains the COBOL FD.

You can enter a string for filtering these names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select names which begin with the letters ABC; %ABC to select names which end with the letters ABC; %ABC% to select names which contain the letters ABC at the beginning, middle, or end.

File Extension

If you selected *Absolute HFS directory pathname* as the file system, enter the extension of the file that contains the COBOL FD.

Select Files and Synonym Names

For each file that you want to map using a COBOL FD, check the box next to the appropriate file name. Select the check box in the table heading to select all files. You can edit the default synonym names. Enter a cluster name to associate it with a particular metadata description. The cluster name is embedded in the Master File. If you do not enter the cluster name during the synonym creation process, you will have to add it dynamically at run time.

Additional Options

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Customize

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Example: Creating a Synonym for a Millennium Data Source

Use the following steps to create a synonym for the DBVSAM01data source:

1. From the Data Adapters page, click *Millennium* and select *Create Synonym* from the pop-up menu. (You can also start from the Metadata page by clicking a Millennium connection.)
2. Enter the following information about the COBOL FD for the file system you use:
   - If you select *Fully qualified PDS name*, enter the name of the PDS that contains the COBOL FD and the member name of the COBOL FD.
   - If you select *Absolute HFS directory pathname* enter the path to the COBOL FD and its file name and extension.
3. Select the COBOL FD to use (in this example DBVSAM01). You can accept the default synonym name, which is the same as the COBOL FD file name, or edit the Default Synonym Name column.
4. Enter the cluster name in the Dataset Location column. For example:

   ```
   user1.dbvsam01.cluster
   ```

5. Click *Create Synonym*.

   You should get a message indicating that the synonym was created successfully.

6. To view the Master File created, go to the Metadata page, open the application under which you created the synonym, click the synonym name, and select *Edit as Text* from the context menu. This example resulted in the following Master File:

   ```
   FILENAME=DBVSAM01, SUFFIX=CPMILL  , $
   DATASET=user1.dbvsam01.cluster, $
     SEGMENT=SEG1, SEGTYPE=S0, $
   $  GROUP=COUNTRY, ALIAS=E1, USAGE=A24, ACTUAL=A24, $
       FIELDNAME=COUNTRY_CODE, ALIAS=E2, USAGE=A8, ACTUAL=A8, $
       FIELDNAME=COUNTRY_NAME, ALIAS=E3, USAGE=A16, ACTUAL=A16, $
   ```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	Definition
On Error	Choose: ❑ *Continue* to continue generating the Master File when an error occurs. Continue is the default value. ❑ *Abort* to stop generating the Master File when an error occurs. ❑ *Comment* to produce a commented Master File when an error occurs.
Hyphens as	Choose: ❑ *No* to remove all hyphens in the COBOL name from the Master File field names. ❑ *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.
Redefines	You may treat COBOL REDEFINE fields in one of three ways. Choose: ❑ *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. ❑ *Comments* to describe REDEFINE fields as comments in the Master File. ❑ *None* to exclude REDEFINE fields altogether.

Parameter	Definition
Occurs as	Choose *Segments* to describe OCCURS structures as segments. Otherwise, choose *Field*. Segments is the default value.
Alignment	Choose: ❑ *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. ❑ *No* to generate Master Files without alignment of slack bytes. No is the default value.
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). ❑ *0* retains the entire COBOL name. *0* is the default value. ❑ *All* removes all prefixes.
Order fields	Choose: ❑ *Yes* to generate Order fields in a Master File. ❑ *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose: ❑ *Comment* to include COBOL Level 88 fields as comments in the Master Files. ❑ *Skip* to exclude level 88 fields. *Skip* is the default value.
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Field Edit Options	

Parameter	Definition
Zeroes	Choose: ❑ *Suppress* to suppress printing of the digit zero for a field whose value is zero. ❑ *Display* to display leading zeroes, for example, 00124. ❑ *None* for no formatting.
Negative value	Choose: ❑ *Bracket* to bracket negative values, for example, (1234). ❑ *Credit* to credit negative values, for example, 1234 CR. ❑ *None* for no formatting.
Dollar Sign	Choose: ❑ *Floating* to display a floating dollar sign and commas, for example, $1,123. ❑ *Fixed* to display a fixed dollar sign and commas, for example, $ 1,123. ❑ *None* for no formatting.
Separate Thousands	Choose: ❑ *Comma* to include commas where appropriate. ❑ *None* for no formatting.

For additional information about customization options, see *Translating COBOL File Descriptions* on page 2699.

Example: **Sample COBOL FD**

```
01 COUNTRY-REC.
   02 G1.
      03  COUNTRY_COD1   PIC X(5).
      03  FILLER         PIC X(3).
    02  COUNTRY_NAM1   PIC X(15).
    02  FILLER         PIC X(1).
```

Standard Master File Attributes for a Millennium Data Source

For each Master File, the server must access the appropriate Millennium database ID (DBID). There are two methods for assigning the DBID to a Master File (see *Specifying the Millennium DBID and TRANID* on page 1577). Depending on your assignment method and on your Millennium lead transaction ID, you may need an Access File in addition to the Master File:

- Master Files describe the fields in Millennium files (see *Master Files* on page 1575). Master Files for Millennium file are the same as Master Files for VSAM files, except that the SUFFIX value for a Millennium file must be CPMILL for Millennium Release 3.

  The Master File must conform to the COBOL copybook for the file. If your site has the COBOL FD Translator feature installed, you can use it to help create a Master File. You can then edit this Master File (to change the SUFFIX value) with any text editor.

- Access Files enable you to use a lead transaction ID (TRANID) other than the default that your site established during installation (see *Specifying the Millennium DBID and TRANID* on page 1577). An Access File can also assign the DBID for its corresponding Master File.

When you issue a report request, the server processes the request with the following steps:

1. It locates the Master File specified by the request.
2. It examines the SUFFIX attribute in the Master File. Each Master File that describes a Millennium database must include the attribute SUFFIX=CPMILL for Millennium Release 3. When the server detects this SUFFIX, it passes control to the adapter.
3. It examines the Master File member name. If an Access File PDS has been allocated, and if it has a member with the same name as the Master File, the adapter locates that Access File.

   The adapter uses the information contained in both the Master File and the Access File (if there is one) to generate the Millennium calls required by the report request. It passes these calls to Millennium.
4. The adapter retrieves the data generated by the Millennium DBMS and returns control to the server. For some requests, the server may perform additional processing on the returned data.

Master Files

The following partial Master File illustrates how to describe a Millennium file. The numbers to the left refer to the explanatory notes that follow the sample:

```
1. FILENAME=ACCTMAST, SUFFIX=CPMILL [,$]
2. SEGNAME=ROOT,SEGTYPE=S0,$
3. FIELDNAME=DELETE_FLAG    ,ALIAS=     ,USAGE=A1  ,ACTUAL=A1  ,$
4. GROUP=CONTRL_KEY         ,ALIAS=KEY  ,USAGE=A23 ,ACTUAL=A23 ,$
       FIELDNAME=CORP       ,ALIAS=     ,USAGE=A3  ,ACTUAL=A3  ,$
       FIELDNAME=ACCOUNT    ,ALIAS=     ,USAGE=A10 ,ACTUAL=A10 ,$
       FIELDNAME=COST_CENTR ,ALIAS=     ,USAGE=A10 ,ACTUAL=A10 ,$
```

Note:

1. Each Master File begins with a file declaration that names the file and describes the type of data source-a Millennium file in this case. The file declaration has two attributes, FILENAME and SUFFIX.

   The FILENAME can be any one- to eight-character name that complies with server naming conventions. For documentation purposes, you can give it the same name as the Master File member name.

   SUFFIX=CPMILL indicates that Millennium Release 3 is required for data retrieval.

2. Each type of Millennium record described in a Master File requires a segment declaration consisting of at least two attributes, SEGNAME and SEGTYPE. The SEGNAME value is ROOT, and the SEGTYPE value is S0 (S zero).

3. To describe a Millennium field in the Master File, you must include a FIELD record (or, for a key field, a GROUP record) that specifies the attributes FIELDNAME, ALIAS, USAGE, and ACTUAL:

   - The FIELDNAME value is the server name for the field. Since field names appear as default column titles on reports, select names that are representative of the data. You can specify field names, aliases, or a unique truncation of either in requests.
   - The ALIAS value in the Master File is an optional alternate name for the field, except if the field is a key field.
   - The USAGE format describes how the server will display the value in reports.
   - The ACTUAL format describes how the data field exists in storage. It must conform to the COBOL FD statement for the field. See ACTUAL Format Conversion Chart for how to determine ACTUAL formats.

4. You must describe the primary key field with the GROUP attribute instead of the FIELDNAME attribute, and you must assign its alias as ALIAS=KEY.

If the key field is composed of multiple elementary fields, describe the elementary fields directly below the GROUP record. The USAGE format of the GROUP must be alphanumeric. Its length is the sum of the server internal storage lengths for the individual fields. The ACTUAL format of the GROUP must also be alphanumeric. Its length is the sum of the subordinate field lengths.

ACTUAL Format Conversion Chart

The ACTUAL attribute indicates the server representation of Millennium field formats. Use the following chart as a guide for describing ACTUAL field formats:

COBOL FORMAT	COBOL PICTURE	FOCUS INTERNAL STORAGE	ACTUAL FORMAT	USAGE FORMAT
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I2
COMP	S9(4)	4	I2	I4
COMP	S9(5	4	I4	I5
COMP	)S9(9	4	I4	I9
COMP-1	)-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	8	P1	P1
COMP-3	S9V99	8	P2	P5.2
COMP-3	9(4)V9(3)	8	P4	P8.3

Note: The USAGE lengths shown are minimum values. You can make them larger and add edit options. You must allow space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits.

Specifying the Millennium DBID and TRANID

You must assign a Millennium database ID (DBID) to each Master File. You can use either of the following techniques to assign the DBID:

- Include the DBID as the last three characters of the Master File member name. For example, if you store the Master File in member ANYGLM, its assigned DBID is GLM (General Ledger).
- Assign the DBID in an Access File. The Access File can also assign the lead transaction ID (TRANID) for its associated Master File.

Create an Access File if:

- The Master File member name does not include a DBID assignment; that is, the last three characters of the Master File member name do not identify a valid DBID.
- The Master File member name assigns a DBID, but the DBID it assigns is not the one you need to access.
- You need a lead transaction ID other than the default (your site establishes the default lead transaction ID during installation).

The Master File member name must either include the appropriate DBID assignment, or it must identify an Access File, or both.

Syntax: How to Specify the Master File Member Name

{*xxxxxdbd*|*anyname*}

where:

xxxxx

Is a name, which can be up to five characters long, that conforms to file-naming conventions.

dbd

Is a 3-character DBID. The adapter accesses this DBID unless an Access File member named xxxxxdbd exists and assigns a different DBID. When both the Master File member name and the Access File assign the DBID, the DBID from the Access File takes precedence.

anyname

Is the name of a member in the Access File data set that contains the DBID and/or lead transaction ID to use.

As an example, consider the Master File in *Master Files* on page 1575. It is member ACCTMAST in the Master File PDS, indicating that member ACCTMAST in the Access File data set contains the required DBID and/or TRANID.

Access Files

If an Access File is required, the server JCL must allocate the PDS containing Access File members to DDNAME FOCPRV.

```
[DBID=dbd], [TRANID={tran|M3LL}]  ,$
```

where:

dbd

Is the three-character Millennium DBID associated with this file. You can omit this parameter if you included the appropriate DBID as the last 3 characters of the Master File member name.

tran

Is the optional Multi:Mill lead transaction ID. M3LL is the default lead transaction ID for Millennium Release 3 unless your site chooses a different default value during installation. For more information, check with your systems support staff.

Note: Every Master File named in a single request (such as a join of two databases) must use the same TRANID.

The following is a sample Access File:

```
DBID=GLM,$
```

This Access File points to:

- A General Ledger file (DBID=GLM).
- The default lead transaction ID, M3LL for Millennium Release 3 (unless the site chose a different default during installation).

Chapter 64

Using the Adapter for Model 204

The Adapter for Model 204 allows applications to access Model 204 data sources. The adapter converts application requests into native Model 204 statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Model 204 Environment

Before you install the adapter, review the following list of software requirements:

- Model 204 must be installed and working. If it is not, contact your Model 204 database administrator. The adapter may be used with Model 204 Version 2.2.0 and higher.
- The server must be installed on your system. If it is not, contact your server administrator or consult the appropriate server installation guide.

Configuring the Adapter for Model 204

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The Adapter for Model 204 can be configured from the Web Console or from the Data Management Console.

Allocating the Model 204 Libraries

Prior to configuring the Adapter for Model 204, you must allocate the following Model 204 libraries.

Edit the ISTART JCL:

- Allocate the Model 204 load library to ddname STEPLIB.
- Any private Master File libraries and *prefix*.EDAMFD.DATA should be allocated to ddname MASTER in the JCL.
- Any private Access File libraries and *prefix*.EDAAFD.DATA should be allocated to ddname ACCESS in the JCL.

Specifying Account and File Passwords

Your Model 204 database administrator must provide account and file security information in order for the adapter to access password protected Model 204 files or groups.

An account consists of the Model 204 user ID, an optional account name, and the associated password. You can specify these security values using the following methods:

- Set appropriate values to declare the CONNECTION attributes and then use the connection name in the Access File CONNECTION attribute.
- Include the ACCOUNT and ACCOUNTPASS attributes in the first declaration of the Access File.
- Issue the adapter M204IN SET M204ACCNT and M204PASS commands.
- Use a security exit. If you use this method, you must omit all previously mentioned sources of account information. For details, see *Using Customized Security Exits* on page 1624.

You must also know the file name and password (if there is one) for each Model 204 file or group you access. Specify these values with the FILE and PASS attributes in each SEGNAME declaration of the Access File.

Testing the Adapter Installation

To verify the adapter installation, run a simple query using one of the pairs of Master and Access Files you have defined for a Model 204 file.

Declaring Connection Attributes

In order to connect to Model 204, the adapter requires connection information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Model 204

The MODEL 204 adapter is under the *DBMS* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Channel Name

Is the CRAM, IUCV, or VMCF communication channel name (up to 8 characters).

Account

Is a string of 1-10 characters, to be used only if an account under which the user will log in to MODEL 204 must be supplied.

Readwol

Readwol (read without locks) instructs the adapter to issue an IFFWOL (find without locks) HLI call. To use this setting, enter ON.

If you leave the setting blank (OFF), the adapter holds locks on the appropriate records in order to ensure accurate data for reports. The locks prevent other users from updating the target records while Model 204 constructs answer sets. The adapter generates standard IFFIND calls that lock the found set in SHR mode.

Singlethread

There are two types (or modes) of thread management:

- **Multiple.** The adapter creates as many threads as it needs whenever they are needed to access files. This setting is recommended for performance reasons.

  To use the default setting (OFF), leave this option blank.

- **Single.** The adapter creates one thread for all file access; this restricts processing to one request. Single thread forces the adapter to use a single thread for all file access. For example, if a request joins 10 files, only one thread is used to access all of the files.

  To use this setting, enter ON.

Missing

You can control the display of Model 204 null data on reports. Null data is translated into the server missing data display value. The default NODATA display value is the period (.).

When the Missing option is selected, the edit specification for all IFGET calls includes an (L) format code. If Model 204 returns a zero in the first byte, the adapter considers that field to be missing. To use this setting, enter ON.

Note: IFFIND calls contain IS PRESENT or IS NOT PRESENT criteria only when the request includes a server IF or WHERE MISSING selection test.

If you leave this setting blank (OFF), null values are not represented on reports. (This is the default setting.)

Note:

- The default setting can affect the results of server DML SUM or COUNT aggregate operations. Null values may be counted or averaged in as existing values.
- You must include the Missing setting in order to specify screening conditions (IF or WHERE MISSING tests) for null values.

Maxmbuf

You can set or change the maximum size of all adapter buffers.

Note that Maxmbuf:

- Should be greater than the larger of two Model 204 buffer settings, LIBUFF and LOBUFF. To identify which buffer is larger, check the parameters in the Model 204 start-up procedure or use the adapter Trace Facility (SET TRACEON=M204IN) to display the settings.
- Should be equal to the size of the largest segment in the Master File if the segment size is greater than the Model 204 LIBUFF and LOBUFF buffer settings. Having the maximum buffer size equal to that of the largest segment prevents S0C4 ABEND conditions.

Enter:

nnnn

To represent the maximum size in bytes. (For example, specify 4096 for a MAXMBUFF value of 4K.) There is no default value.

Ftblsize

The Ftblsize option enables you to control the size of the Model 204 FTBL buffer. The LFTBL parameter, specified in the Model 204 start-up procedure, governs the size of the FTBL buffer.

Enter:

nnnn

To represent the FTBL size in bytes. (For example, specify 4096 for an FTBL value of 4K.)

You should set FTBL to a value larger than the previously defined Model 204 LFTBL value.

User

Is the user ID (1-10 characters) by which you are known to MODEL 204. If omitted, other sources of this data will be used (that is, file level information from the Access File or a security exit).

Password

Is the password associated with the user ID. If omitted, other sources of this data will be used (that is, file level information from the Access File or a security exit)

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE M204INX SET CONNECTION_ATTRIBUTES connection_name
channel_name/userid,password:
"[ACCOUNT account] [READWOL {ON|OFF}] [SINGLETHREAD {ON|OFF}]
[MISSING {ON|OFF}] [MAXMBUF nnnnn] [FTBLSIzE nnnnn]"]
```

where:

connection_name

Is the name of the current connection.

channel_name

Is the CRAM, IUCV, or VMCF communication channel name (up to 8 characters).

`userid,password`

Are valid M204 user IDs and passwords. If they are omitted then other sources of this data will be used (that is, file level information from the Access File or a security exit).

`account`

Is a string of 1-10 characters, to be used only if an account under which the user will log in to Model 204 must be supplied.

A space is used as the delimiter between the parameters and values.

`READWOL`

The options are:

`ON` instructs the adapter read without locks and issue an IFFWOL (find without locks) HLI call.

`OFF` holds locks on the appropriate records to ensure accurate data for reports. The locks prevent other users from updating the target records while Model 204 constructs answer sets. The adapter generates standard IFFIND calls that lock the found set in SHR mode. (This is the default setting.)

`SINGLETHREAD`

The options are:

`ON` is used for single thread processing. With this setting, the adapter creates one thread for all file access; this restricts processing to one request. For example, if a request joins 10 files, only one thread is used to access all of the files.

`OFF` is used for multiple thread processing. With this setting, the adapter creates as many threads as it needs whenever they are needed to access files. OFF, the default setting, is recommended for performance reasons.

`MISSING`

You can control the display of Model 204 null data on reports. Null data is translated into the server missing data display value. The default NODATA display value is the period (.).

The options are:

`ON` where in this setting, the edit specification for all IFGET calls includes an (L) format code. If Model 204 returns a zero in the first byte, the adapter considers that field to be missing.

Note that IFFIND calls contain IS PRESENT or IS NOT PRESENT criteria only when the request includes a server IF or WHERE MISSING selection test.

OFF where in this setting, null values are not represented on reports. This is the default setting.

Note: The default setting can affect the results of server DML SUM or COUNT aggregate operations. Null values may be counted or averaged in as existing values.

You must include the Missing setting in order to specify screening conditions (IF or WHERE MISSING tests) for null values.

MAXMBUF

You can set or change the maximum size of all adapter buffers.

nnnn represents the maximum size in bytes. (For example, specify 4096 for a MAXMBUFF value of 4K.) There is no default value.

Note that MAXMBUF:

- Should be greater than the larger of two Model 204 buffer settings, LIBUFF and LOBUFF. To identify which buffer is larger, check the parameters in the Model 204 start-up procedure or use the adapter Trace Facility (SET TRACEON=M204IN) to display the settings.
- Should be equal to the size of the largest segment in the Master File if the segment size is greater than the Model 204 LIBUFF and LOBUFF buffer settings. Having the maximum buffer size equal to that of the largest segment prevents S0C4 ABEND conditions.

FTBLSIZE

The FTBLSIZE option enables you to control the size of the Model 204 FTBL buffer. The LFTBL parameter, specified in the Model 204 start-up procedure, governs the size of the FTBL buffer.

nnnn represents the FTBL size in bytes. (For example, specify 4096 for an FTBL value of 4K.)

Note that you should set FTBL to a value larger than the previously defined Model 204 LFTBL value.

Any of the environmental SET commands in the prior version of Model 204 may be used, except M204ACCNT and M204PASS.

Example: Sample Connection Declarations

```
ENGINE M204INX SET CONNECTION_ATTRIBUTES m204a IFAMCHNL/user1,
pwd1: "ACCOUNT acc1 MISSING ON READWOL ON FTBLSIZE 16000"
```

```
ENGINE M204INX SET CONNECTION_ATTRIBUTES m204b IFAMCHNL/user2,
pwd2: "SINGLETHREAD ON"
```

The command ? SET may point to a valid connection_name. For example:

```
ENGINE M204INX SET CONNECTION_ATTRIBUTES M204
IFAMSUPT/SUPERKLUGE,PIGFLOUR: "ACCOUNT ACT1 MISSING ON"

ENGINE M204INX ? SET
(FOC4855) Set M204ACCNT option
(FOC4856) Set M204PASS option
(FOC4857) Set SINGLETHREAD option   : OFF
(FOC4858) Set READWOL option        : OFF
(FOC4859) Set MISSING option        : OFF
(FOC4860) Set MAXMBUFF option
(FOC4861) Set FTBLSIZE option
ENGINE M204INX ? SET M204
(FOC4863) Set IFAMCHNL option       : IFAMSUPT
(FOC4855) Set M204ACCNT option      : SUPERKLUGE ACT1
(FOC4856) Set M204PASS option       : PIGFLOUR
(FOC4857) Set SINGLETHREAD option   : OFF
(FOC4858) Set READWOL option        : OFF
(FOC4859) Set MISSING option        : ON
(FOC4860) Set MAXMBUFF option       : 0
(FOC4861) Set FTBLSIZE option       : 0
```

Model 204 Overview and Mapping Considerations

A Model 204 file can represent one file or a collection of files called a group. A single Model 204 file can contain records that vary in length and format; a record can consist of fields that also vary in length. Fields are the smallest elements or units of data. They can occur more than once per record. A unit of records with identical sets of fields is called a logical record type. A Model 204 file is defined in a Model 204 file description.

In the sections that follow, you will notice that:

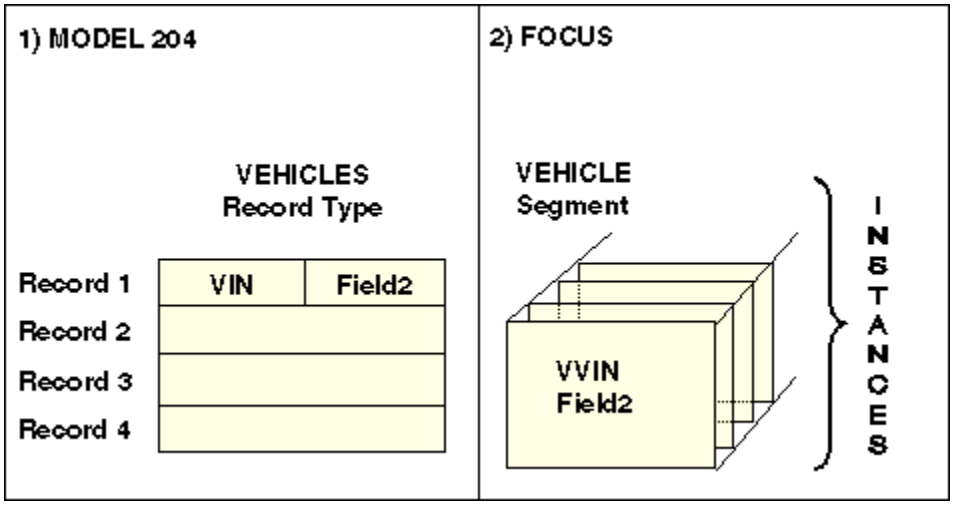
- A Model 204 field is equivalent to a server Master File field.
- A logical record type corresponds to a server Master File segment. The individual record corresponds to a segment instance.
- Model 204 fields that appear more than once in a record map to a server Master File OCCURS segment.
- One or more record types can be described in any order in a pair of Master and Access Files.
- One or more Model 204 files can be described in a pair of Master and Access Files.

Files With One Logical Record Type

A Model 204 file that contains records with identical sets of fields (the same logical record type) can be represented as a single Master File segment. Each Model 204 field that occurs only once becomes a Master File field. The ALIAS attribute in the Master File identifies the Model 204 field name.

In the Master File, describe the logical record type as a segment. The segment description can include some or all of the Model 204 fields in any order. The corresponding Access File identifies the record type's Model 204 file or group name and its password; the Access File can also include FIELD declarations that specify a TYPE attribute to indicate the appropriate suffix operators for Model 204 key fields. Master File field suffix operators (for example, KEY) identify different types of Model 204 key fields.

Note: A Model 204 field that can occur more than once should be represented as a separate OCCURS segment.



1. Is one Model 204 file with one logical record type.
2. Is the equivalent server Master File segment. The Master File field VVIN represents the Model 204 field VIN.

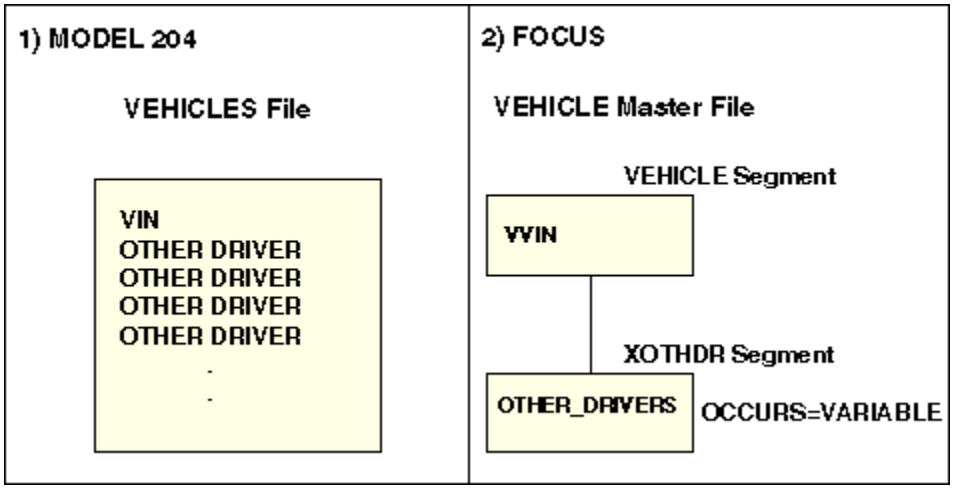
Files With Multiply Occurring Fields

In Model 204 files, some fields may appear more than once per record. Represent each multiply occurring field separately from the non-repeating fields as a server Master File OCCURS segment. In this case, the Master File contains a parent segment to describe all the non-repeating fields, and a separate dependent OCCURS segment for each multiply occurring field.

When you create a Master File, describe the OCCURS segment only if you need the multiply occurring field for your reports. You are not required to describe every multiply occurring field. Do not describe the OCCURS segment in the corresponding Access File unless its size exceeds the buffer capacity, a situation that is explained in *Customizing the Model 204 Environment* on page 1619.

Describing Files to the Server

The following diagram illustrates one multiply occurring field and its equivalent OCCURS segment:



1. Is the Model 204 VEHICLES file with one logical record type and one multiply occurring field called OTHER DRIVER that identifies drivers besides the principal driver.
2. Is the equivalent multi-segment server structure. The parent (or root, in this case) is the VEHICLE segment; it contains all of the non-repeating fields. The dependent segment is the OCCURS segment which contains iterations of the Model 204 OTHER DRIVER field. In the Master File, the segment declaration for the dependent segment must include the OCCURS attribute.

If a Model 204 file contains multiple logical record types, each logical record type corresponds to one segment. You can represent a Model 204 file with several logical record types either:

- As a multi-segment server Master File structure with a hierarchical retrieval path. The segments are cross-referenced by Embedded Joins defined on the parent/child relationships.
- As several individual segments, each described in a separate pair of Master and Access Files.

In a multi-segment Master File, you can include segment descriptions for all of the logical record types or only those you need for your reports. To identify the parent of a dependent segment, specify the PARENT attribute in the segment declaration of the dependent segment.

In the corresponding Access File, for each logical record type you must specify the:

- Model 204 file and password.
- Model 204 record type field and the unique value that identifies it.
- Shared field that implements the Join relationship.
- Model 204 key fields (TYPE attributes in FIELD declarations identify the appropriate suffix operators).

Note: Record type fields identify individual record types that reside in the same Model 204 file. Model 204 files that consist of one logical record type do not require record type fields. In the Access File, the RECTYPE and RECTVAL attributes specify record type fields and the values.

Mapping Model 204 and Server Relationships

In Model 204, interfile relationships are not coded in the Model 204 file description. This provides a certain degree of flexibility because the Model 204 files can be cross-referenced dynamically. On the other hand, the burden of connecting the Model 204 files falls on the user.

Model 204 relationships between logical record types are cross-referenced using shared key fields. The adapter supports this implementation. You can logically join Model 204 files using the adapter with either of the following techniques:

- Issue JOIN commands to dynamically join the Model 204 files.
- Create a multi-segment server Master File structure and describe the Join relationships in a pair of Master and Access Files (this multi-segment structure is also called an Embedded Join or Server View).

Both techniques are described in the following sections.

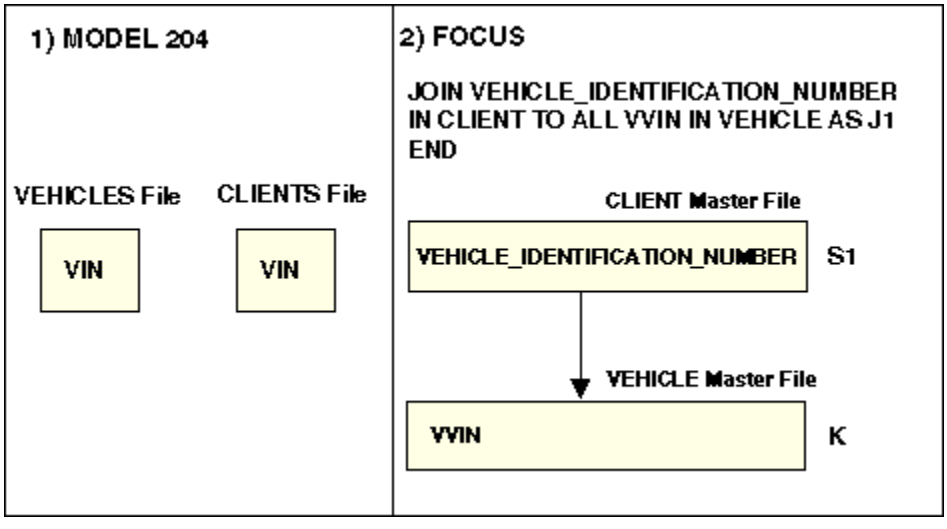
Dynamic Joins

If there is a shared or common field, you can use the server JOIN command to dynamically Join:

- Separate physical Model 204 files that each contain one logical record type. These Model 204 files are usually described as single segment Master Files.

- Logical record types that are described to the server as single segment Master Files, but that originate from the same Model 204 file.
- Multi-segment server structures that represent several Model 204 files and/or different record types from the same Model 204 file. Multi-segment Master Files are explained in *Embedded Joins* on page 1592.

The following diagram illustrates a Model 204 Join and its equivalent Server Join:

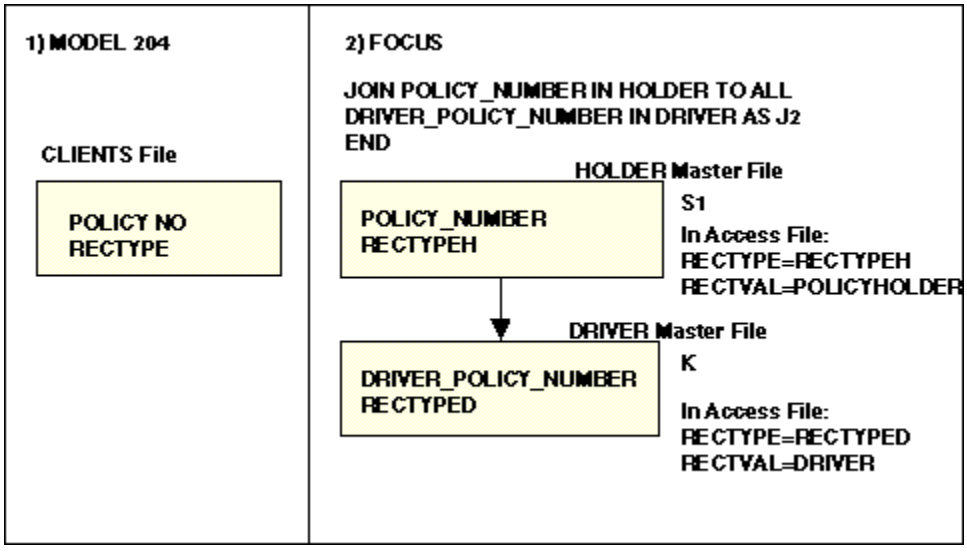


1. Are two Model 204 files, VEHICLES and CLIENTS. Each file contains one logical record type for this example. The common or key field is VIN.
2. Are two equivalent server segments. Each segment is described in a separate pair of Master and Access Files. The field names for the common field are VEHICLE_IDENTIFICATION_NUMBER and VVIN. The Model 204 key fields are specified with the TYPE=KEY attribute in the Access File.

The JOIN command includes the names of the Master Files and the common field names. To retrieve data from the joined structure, issue a report request that specifies the host (or parent) Master File from the JOIN command.

Note: Since in this example each Model 204 file consists of one logical record type, record type fields are not specified in the Access Files.

The following diagram illustrates a Model 204 file with two record types and the Server Join:



1. Is the Model 204 CLIENTS file containing two logical record types. The common or key field is POLICY NO. The RECTYPE field contains the value POLICYHOLDER for the HOLDER record type and the value DRIVER for the DRIVER record type.
2. Are two equivalent Master File segments. Each segment is described in a separate pair of Master and Access Files. The field names for the common field are POLICY_NUMBER and DRIVER_POLICY_NUMBER. The Model 204 key fields are specified with the TYPE=KEY attribute in the Access File. Since both record types reside in the same Model 204 file, each Access File segment declaration also includes RECTYPE (record type) and RECTVAL (record type value) attributes.

The JOIN command includes the names of the Master Files and the common field names. To retrieve data from the joined structure, issue a report request that specifies the host (or parent) Master File from the JOIN command.

Embedded Joins

You can also implement Join relationships by describing a multi-segment FOCUS structure in a single pair of Master and Access Files. If there is a shared or common field, you can create an Embedded Join for:

- Separate physical Model 204 files that each contain one logical record type. The Model 204 files are described as segments in the multi-segment structure.

- Logical record types that originate from the same Model 204 file. These record types are described as segments in the multi-segment structure. In the Access File, the RECTYPE and RECTVAL attributes in each segment record identify the record type field and its corresponding value.
- Other multi-segment FOCUS structures. A Master File might relate two hierarchical structures as one. For example, the AUTO Master File might incorporate the VEHICLE and CLIENT Master Files.

Embedded Joins offer two advantages:

- They create a logical FOCUS view of the record types. The view may also provide a measure of security by limiting access to segments or fields.
- You do not have to issue an explicit JOIN command.

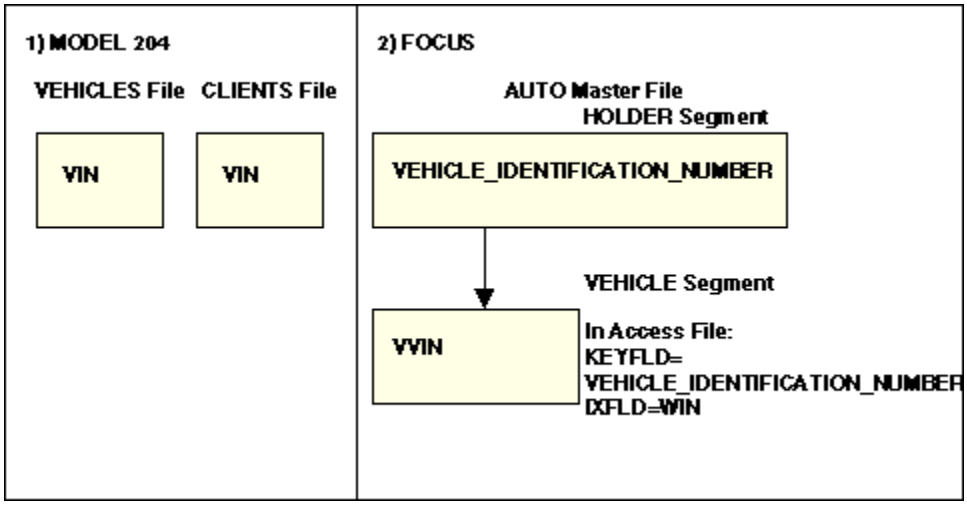
When you create a Master File, describe the Model 204 logical record types or files as a hierarchy. Start first by choosing a record type or file to be the root segment. The root segment has dependent segments which, in turn, may have dependent segments. The dependent segments identify the superiors (or parents) with the PARENT attribute. Any record type can act as a dependent provided that its shared field is a key field.

In the associated Access File, for each pair of related segments, specify the shared field from the parent and dependent segments with the KEYFLD and IXFLD attributes. The Interface implements the relationship by matching values at run time.

The Embedded Join is executed automatically when you issue a report request that references fields from two related segments. You do not specify the shared field in your report request; the selection of the shared field is transparent.

Note: It is possible to join unrelated Model 204 files and record types that do not have common fields by describing a DUMMY segment.

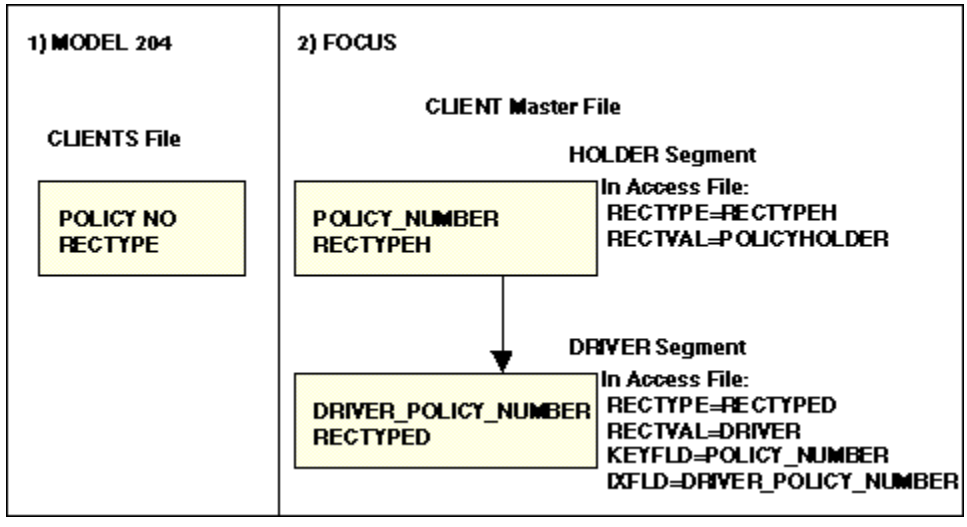
The following diagram illustrates an Embedded Join for two Model 204 files:



1. Are two Model 204 files, VEHICLES and CLIENTS. Each file contains one logical record type for this example. Since they exist as physically separate files, they do not contain a Model 204 record type field. The common or key field is VIN.
2. Are two equivalent FOCUS segments described in one pair of Master and Access Files. The field names for the common field are VEHICLE_IDENTIFICATION_NUMBER and VVIN. The Model 204 key fields are specified with the TYPE=KEY attribute in the Access File.

   To create a parent/child relationship, describe the HOLDER segment as the parent (or root) and the VEHICLE segment as the dependent in the Master File. Include the PARENT=HOLDER attribute in the segment declaration for VEHICLE. In the Access File, specify the KEYFLD and IXFLD attributes to create the Embedded Join.

The following diagram illustrates an Embedded Join for a Model 204 file with two record types:



1. Is the Model 204 CLIENTS file containing two logical record types. The common or key field is POLICY NO. The RECTYPE field contains the value POLICYHOLDER for the HOLDER record type and the value DRIVER for the DRIVER record type.
2. Are two equivalent FOCUS segments described in one pair of Master and Access Files. The field names for the common field are POLICY_NUMBER and DRIVER_POLICY_NUMBER. The Model 204 key fields are specified with the TYPE=KEY attribute in the Access File. Since both record types reside in the same Model 204 file, the Access File segment declarations also include the RECTYPE (record type) and RECTVAL (record type value) attributes.

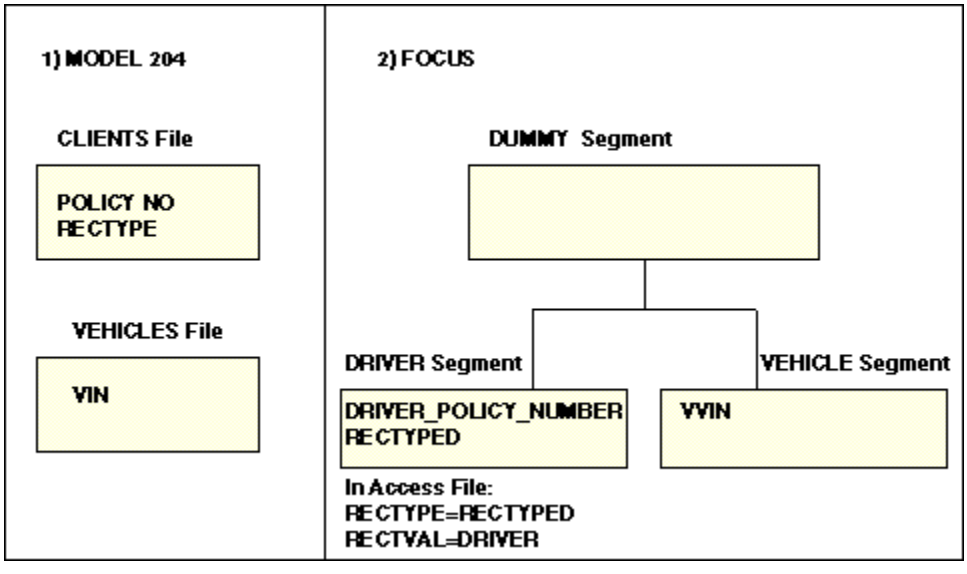
   To create a parent/child relationship, describe the HOLDER segment as the parent (or root) and the DRIVER segment as the dependent in the Master File. Include the PARENT=HOLDER attribute in the segment declaration for DRIVER. In the Access File, specify the KEYFLD and IXFLD attributes in the DRIVER segment declaration to create the Embedded Join.

Joining Unrelated Files

You can describe unrelated Model 204 files or logical record types as Embedded Joins even if a shared field does not exist. To do so, describe a special dummy segment (SEGNAME=DUMMY) as the root in the Master File. Then, specify the unrelated segments as parallel dependents of the dummy root. Since the dummy segment is a virtual segment, do not describe fields for it in the Master File, and do not include a corresponding segment declaration for it in the Access File.

If your report request includes fields from two or more segments, FOCUS makes only one retrieval pass over the answer set returned by the Model 204 DBMS. Fields specified for sort phrases (BY and ACROSS) and screening tests (IF and WHERE) must lie on the same retrieval path as the other requested fields. You cannot use a field from one unrelated segment to sort the data in other segments.

The following diagram illustrates two Model 204 files, the CLIENTS file and the VEHICLES file. These two files are joined using a FOCUS dummy segment:



1. Are two Model 204 files, CLIENTS and VEHICLES. The record type field in the DRIVER segment identifies all client records that have the value 'DRIVER'. A dummy segment is needed because there is no shared field to cross-reference or join these files.
2. Is an equivalent multi-segment FOCUS structure. The root is a DUMMY segment. The two dependents are related to the root. They are not related to each other because a common field does not exist.

To create a Master File for this structure, specify the segment declaration for the dummy segment (SEGNAME=DUMMY) first, so it functions as the root. Do not describe fields for it. Then, describe the dependent segments, and include the PARENT=DUMMY attribute to identify the dummy segment as the parent. In the associated Access File, do not specify a segment declaration for the dummy segment; include segment declarations only for the dependent segments. The record type field is required in the DRIVER segment since the DRIVER segment comes from a Model 204 file, CLIENTS, that contains two logical record types. Do not specify KEYFLD/IXFLD values.

Summary of Mapping Rules

Model 204 and FOCUS use similar elements:

Model 204 Entities	FOCUS Entities
Model 204 field	FOCUS field
Logical record type	FOCUS segment
Individual record	Segment instance
Model 204 file with one record type	FOCUS segment
Multiply-occurring Model 204 fields in a record	FOCUS OCCURS segment

Some general rules for describing Model 204 files are:

- Each logical record type becomes a FOCUS segment. A FOCUS segment may represent an entire Model 204 file (with one logical record type) or one logical record type from a Model 204 file with several record types.
- A Model 204 file containing two or more logical record types must have a record type field to identify each record type.
- A Model 204 file containing one logical record type does not require a record type field; the Model 204 file may contain an optional one.
- Relationships can be implemented with the JOIN command or by describing Embedded Joins in the FOCUS file descriptions. The maximum number of Model 204 files or logical record types that can be related is 64 for Embedded Joins and 16 for Dynamic Joins.

Managing Model 204 Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Model 204 data types.

The logical description of a Model 204 file is stored in a Master File, which describes the field layout. The physical attributes of the Model 204 file, such as the keys description, occurring data access, and security are stored in the Access File.

Creating Synonyms

Synonyms define unique names (or aliases) for each Model 204 table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Model 204

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

File name

Is the Model 204 file name.

File password

Is the associated password for the file. This entry is optional.

Rectype name

Is a valid Model 204 field with the name RECTYPE. This entry is optional.

Rectype value

Is a valid value of the Model 204 field with the name RECTYPE. This entry is optional.

Synonym name

Displays the name that will be assigned to the synonyms. To assign a different name, replace the displayed value.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Example: Creating a Synonym

To generate a synonym for the VEHICLES data source, enter the following information on the Create Synonym panes of the Web Console or the Data Management Console:

1. On the first Create Synonym pane, enter VEHICLES in the File Name field and WHEELS in the File Password field.
2. Enter *m204veh* as the Synonym name.
3. Select *m204* from the Application list.
4. Click *Create Synonym*. The synonym is created and added under the specified application directory (baseapp is the default).
5. The Create Synonym Status pane indicates that the synonym was created successfully. Click *Applications* on the menu bar.
6. Open the *baseapp* application folder in the navigation pane and click the synonym *m204*.
7. Choose *Edit as Text* from the menu to view (and edit) the generated Master File and Access File.

Generated Master File:

```
FILENAME=M204VEH, SUFFIX=M204INX , $
  SEGMENT=VEHICLES, SEGTYPE=S0, $
    FIELDNAME=BODY, ALIAS=BODY, USAGE=A4, ACTUAL=A4, $
    FIELDNAME=COLLISION_PREMIUM, ALIAS='COLLISION PREMIUM', USAGE=I9, ACTUAL=A9, $
    FIELDNAME=COLOR, ALIAS=COLOR, USAGE=A255, ACTUAL=A255, $
    FIELDNAME=DEDUCTIBLE, ALIAS=DEDUCTIBLE, USAGE=I9, ACTUAL=A3, $
    FIELDNAME=GARAGING_LOCATION, ALIAS='GARAGING LOCATION', USAGE=A4, ACTUAL=A4, $
    FIELDNAME=LIABILITY_LIMIT, ALIAS='LIABILITY LIMIT', USAGE=I9, ACTUAL=A3, $
    FIELDNAME=LIABILITY_PREMIUM, ALIAS='LIABILITY PREMIUM', USAGE=I9, ACTUAL=A9, $
    FIELDNAME=MAKE, ALIAS=MAKE, USAGE=A255, ACTUAL=A255, $
    FIELDNAME=MODEL, ALIAS=MODEL, USAGE=A255, ACTUAL=A255, $
    FIELDNAME=OTHER_DRIVER, ALIAS='OTHER DRIVER', USAGE=A255, ACTUAL=A255, $
    FIELDNAME=OWNER_POLICY, ALIAS='OWNER POLICY', USAGE=A6, ACTUAL=A6, $
    FIELDNAME=PRINCIPAL_DRIVER, ALIAS='PRINCIPAL DRIVER', USAGE=I9, ACTUAL=A9, $
    FIELDNAME=SURCHARGE%, ALIAS=SURCHARGE%, USAGE=A255, ACTUAL=A255, $
    FIELDNAME=TRANS, ALIAS=TRANS, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=USAGE, ALIAS=USAGE, USAGE=A255, ACTUAL=A255, $
    FIELDNAME=VEHICLE_PREMIUM, ALIAS='VEHICLE PREMIUM', USAGE=I9, ACTUAL=A9, $
    FIELDNAME=VEHICLE_RATING, ALIAS='VEHICLE RATING', USAGE=A1, ACTUAL=A1, $
    FIELDNAME=VEHICLE_USE_CLASS, ALIAS='VEHICLE USE CLASS', USAGE=A2, ACTUAL=A2, $
    FIELDNAME=VIN, ALIAS=VIN, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=YEAR, ALIAS=YEAR, USAGE=A255, ACTUAL=A255, $
```

Generated Access File:

```
SEGNAM=VEHICLES, CONNECTION=M204, FILE=VEHICLES, PASS=WHEELS, $
 FIELD=BODY, TYPE=ORA, $
 FIELD=COLLISION_PREMIUM, TYPE=ORN, $
 FIELD=COLOR, TYPE=ORA, $
 FIELD=DEDUCTIBLE, TYPE=ORN, $
 FIELD=GARAGING_LOCATION, TYPE=ORA, $
 FIELD=LIABILITY_LIMIT, TYPE=ORN, $
 FIELD=LIABILITY_PREMIUM, TYPE=ORN, $
 FIELD=MAKE, TYPE=ORA, $
 FIELD=MODEL, TYPE=ORA, $
 FIELD=OTHER_DRIVER, TYPE=KEY, $
 FIELD=OWNER_POLICY, TYPE=ORA, $
 FIELD=PRINCIPAL_DRIVER, TYPE=ORN, $
 FIELD=TRANS, TYPE=ORA, $
 FIELD=USAGE, TYPE=ORA, $
 FIELD=VEHICLE_PREMIUM, TYPE=ORN, $
 FIELD=VEHICLE_RATING, TYPE=ORA, $
 FIELD=VEHICLE_USE_CLASS, TYPE=ORA, $
 FIELD=VIN, TYPE=ORA, $
```

M204 fields may have KEY and ORDERED attributes at the same time. In such cases, the field in the Access File has TYPE=KEY(IVK for invisible field) and a comment with information about another possible TYPE:

```
 SEGNAM=RNG, CONNECTION=M204, FILE=RNG, $
  FIELD=FKEY, TYPE=KEY, ATTRIBUTE=FRV, $
  FIELD=FORC, TYPE=KEY, $
$  Field has KEY and ORC attributes. KEY is chosen.      $
  FIELD=FORN, TYPE=KEY, $
$  Field has KEY and ORN attributes. KEY is chosen.      $
  FIELD=FRNG, TYPE=RNG, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Master Files for Model 204

A Model 204 file consists of records and fields. It can represent one file or a collection of files called a group. If records have identical sets of fields, the unit of records is called a logical record type. A logical record type is described as a single segment in the Master File.

A Master File consists of file, segment, and field declarations. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$). Text appearing after the comma and dollar sign is treated as a comment.

- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.
- Certain attributes are required; the rest are optional.

The following sections summarize the syntax for each type of declaration and then present detailed explanations for each attribute.

Syntax: How to Declare File Attributes

Each Master File begins with a file declaration that names the file and describes the type of data source, a Model 204 file or group in this case. The file declaration has two attributes, FILENAME and SUFFIX

```
FILE[NAME]=name, SUFFIX=M204IN [,$]
```

where:

name

Is a 1- to 8-character name that identifies the Master File. The name must comply with server file naming conventions.

M204IN

Is the SUFFIX value that specifies the Adapter for Model 204.

Syntax: How to Declare Segment Attributes

Each logical record type described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE

```
SEGNAME={name|DUMMY}, SEGTYPE={S0|U} [, PARENT=parent] ,$
```

where:

name

Is a 1- to 8-character name that identifies the segment. The value DUMMY creates a virtual segment used for joining unrelated segments.

S0

Indicates to the adapter that the Model 204 DBMS handles the storage order of the data. S0 is the default setting.

parent

For multi-segment structures (Embedded Joins) only, is the SEGNAME value for the parent record type. This attribute is required in segment declarations that describe dependent segments.

SEGNAME

The SEGNAME (or SEGMENT) attribute identifies each Model 204 logical record type. The SEGNAME value can be a maximum of 8 alphanumeric characters. It may be the name of the record type or an arbitrary name and must be unique within a Master File.

The SEGNAME value from the Master File is also specified in the Access File where the segment declaration identifies the name of the Model 204 file. In this manner, the SEGNAME value serves as a link to the actual Model 204 file name.

SEGTYPE

In a single segment Master File, the SEGTYPE value is S0. The Model 204 DBMS handles the storage order of the data.

In a multi-segment Master File, SEGTYPE values for dependent segment declarations can be S0 or U (for unique).

- SEGTYPE=S0 indicates that the dependent record type has a one-to-many or many-to-many relationship with the record type named as its parent. For every record of the parent record type, there may be more than one record in the dependent record type.
- SEGTYPE=U indicates that the dependent record type has a one-to-one or a many-to-one relationship with the record type named as its parent. For every record of the parent record type, there may be (at most) one record in the dependent record type. For a one-to-one relationship to exist, both record types must share a common key. For a many-to-one relationship to exist, the common key of the dependent record type must be a subset of the common key of the parent record type.

If the SEGTYPE attribute is omitted from any segment record in the Master File, its value defaults to S0.

PARENT

The PARENT attribute is required in dependent segment declarations. Its value is the SEGNAME value of the dependent record type's parent (the logical record type to which it will be related at run time).

Syntax: How to Declare Field Attributes

Each record in a logical record type consists of one or more fields. These Model 204 fields are described in the Master File with the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL. You are only required to describe Model 204 fields you use in reports, and you can specify them in any order within each segment

```
FIELD[NAME]=field, [ALIAS=] [m204field], [USAGE=]display, [ACTUAL=]An ,$
```

where:

field

Is a 1- to 66-character field name. In requests, the field name can be qualified with the Master File and/or segment name. Although the qualifiers and qualification characters do not appear in the Master File, they count toward the 66 character maximum.

m204field

Is the Model 204 field name, which can be up to 66 characters (or, optionally left blank if the field name exceeds 66 characters and is described in the Access File).

display

Is the server display format for the field.

A*n*

Is the server definition of the Model 204 field format and length (*n*).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=YEAR, USAGE=A2, ACTUAL=A2,$
FIELD = YEAR, YEAR, A2, A2,$
```

FIELDNAME

Field names must be unique within the Master File and may consist of up to 66 alphanumeric characters. Model 204 field names are acceptable values if they meet the following naming conventions:

- Names can consist of letters, digits, and underscore characters. Special characters and embedded blanks are not advised.
- The name must contain at least one letter.

Since the field name appears as the default column title for reports, select a name that is representative of the data. You can specify field names, aliases, or a unique truncation of either in requests.

Duplicate field names (the same field names and aliases) within a segment are not permitted. Requests can qualify duplicate field names from different segments with the name of the Master File and/or segment.

Note: Field names specified in OCCURS segments may not be qualified.

ALIAS

The ALIAS value for each field must be the full Model 204 field name; the adapter uses it to generate HLI calls. The ALIAS name must be unique within the segment. If the name is longer than the 66 character maximum, leave the ALIAS attribute blank in the Master File and specify the alias in the Access File instead. The ALIAS name must comply with the field naming conventions listed previously. If the Model 204 field name contains an embedded blank, enclose the name in single quotation marks.

Aliases may be duplicated within the Master File if they are defined for different Model 204 logical record types. However, the corresponding field names must be unique. For example, the ALIAS values for two Model 204 record type fields may be RECTYPE, but the field names must be unique.

Note: Do not use RECTYPE as a field name.

USAGE

The USAGE attribute indicates the display format of the field. An acceptable value must include the field type and length and may contain edit options. The server uses the USAGE format for data display on reports. All standard server USAGE formats (A, D, F, I, P) are available.

ACTUAL

The ACTUAL attribute indicates the server representation of Model 204 field formats as returned by the Model 204 DBMS in the data buffer. Specify an alphanumeric format (A) with a length (n) equal to the maximum length for the Model 204 field. If the ACTUAL length is less than the maximum length for the Model 204 field, field values will be truncated.

Note: Since Model 204 file descriptions do not define field lengths, ask your Model 204 database administrator or consult other sources, such as data dictionaries, for field information.

Syntax: How to Declare OCCURS Attributes

Describe each multiply occurring field that exists in a Model 204 record as an OCCURS segment. The segment declaration for an OCCURS segment consists of the SEGNAME, PARENT, and OCCURS attributes. It contains one field declaration to represent the multiply occurring field, and, optionally, a field declaration for an internal counter, the ORDER field. Each OCCURS segment is described as a dependent segment. Its parent segment describes all non-repeating fields from the logical record type. You do not have to describe an OCCURS segment if you do not plan to use the multiply occurring field in report requests

```
SEGNAME=segname, PARENT=parentname, OCCURS=VARIABLE,$
```

where:

segname

Is the 1- to 8-character name of the OCCURS segment.

parentname

Is the SEGNAME value of the parent segment that contains the non-repeating fields from the logical record type.

VARIABLE

Indicates that the number of occurrences varies.

Note:

- The SEGTYPE attribute is always S0 for OCCURS segments and, therefore, can be omitted.
- OCCURS segments generally do not require corresponding segment declarations in the Access File. An exception is for processing OCCURS segments that exceed the buffer capacity.
- Field names in OCCURS segments may not be qualified with Master File or segment names, but the TYPE attribute can define suffix operators in the Access File.

Syntax: How to Track a Sequence Within Multiply Occurring Fields

The ORDER field is an optional counter that you can define to identify the sequence number of each multiply occurring field when the order of data is significant. You can use its value in subsequent report requests. The ORDER field does not represent an existing Model 204 field; it is used only for internal processing. The ORDER field must be the last field described in the OCCURS segment

```
FIELD=name, ALIAS=ORDER, USAGE=In, ACTUAL=I4,$
```

where:

name

Is any meaningful name.

ALIAS

Must be ORDER.

USAGE

Is an integer (I*n*) format.

ACTUAL

Must be I4.

The following annotated example illustrates an OCCURS segment in the VEHICLE Master File.

```
1. SEGNAME=VEHICLE,$
     FIELD=VVIN ,ALIAS=VIN  ,A10 ,A10 ,$
     FIELD=YEAR ,ALIAS=YEAR ,A2  ,A2  ,$
     FIELD=MAKE ,ALIAS=MAKE ,A16 ,A16 ,$
2. SEGNAME=XOTHDR, PARENT=VEHICLE, OCCURS=VARIABLE ,$
3.   FIELD=OTHER_DRIVERS, ALIAS='OTHER DRIVER' ,A6 ,A6 ,$
4.   FIELD=DRVCNT, ALIAS=ORDER ,I4 ,I4 ,$
```

Notice that:

1. Is the segment declaration for the parent segment, VEHICLE.
2. Is the OCCURS segment declaration.
3. Is the field declaration for the multiply occurring field. Field qualifiers (Master File and segment names) are not permitted, but the TYPE attribute can define a suffix operator in the Access File.
4. Is the field declaration for the ORDER field. As an internal counter field, it does not correspond to a Model 204 field.

Access Files for Model 204

Each Master File for a Model 204 file must have a corresponding Access File. The name of the Access File must be the same as that of the Master File. In z/OS, the Access File PDS is allocated to ddname ACCESS in the server JCL. The Access File associates a segment in the Master File with the Model 204 logical record type it describes.

The Access File consists of account, segment, and field declarations. The Access File minimally identifies account and segment information and field suffix operators. It can also contain field declarations for aliases that exceed 66 characters.

In multi-segment structures, the Access File must contain a segment declaration for each logical record type described in the Master File. Segment information may include Embedded Joins and Model 204 record type fields. The order of segment declarations in the Access File must correspond to the order in the Master File.

The Access File consists of 80 character declarations in comma-delimited format. Rules for declarations are:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,$).
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas separate attribute=value pairs.
- Blank lines and leading or trailing blanks around attribute=value pairs are ignored.
- Declarations starting with an asterisk (*) in Column 1 are treated as comments.
- Values that contain commas, equal signs, dollar signs, or embedded blanks must be enclosed in single quotation marks.
- If you wish, you can number the declarations in columns 73 through 80.

For example, the VEHICLE Access File contains one segment declaration because the VEHICLE Master File contains only one logical record type.

```
FILE=VEHICLE1, SUFFIX=M204IN

SEGNAME=VEHICLE,$
FIELD=VVIN                ,ALIAS=VIN                  ,A10   ,A10   ,$
FIELD=YEAR                ,ALIAS=YEAR                 ,A2    ,A2    ,$
FIELD=MAKE                ,ALIAS=MAKE                 ,A16   ,A16   ,$
FIELD=MODEL               ,ALIAS=MODEL                ,A12   ,A12   ,$
FIELD=BODY                ,ALIAS=BODY                 ,A4    ,A4    ,$
FIELD=COLOR               ,ALIAS=COLOR                ,A10   ,A10   ,$
FIELD=VEHICLE_PREMIUM     ,ALIAS='VEHICLE PREMIUM'    ,A8    ,A8    ,$
FIELD=COLLISION_PREMIUM   ,ALIAS='COLLISION PREMIUM'  ,A6    ,A6    ,$
FIELD=LIABILITY_PREMIUM   ,ALIAS='LIABILITY PREMIUM'  ,A12   ,A12   ,$
FIELD=LIABILITY_LIMIT     ,ALIAS='LIABILITY LIMIT'    ,A3    ,A3    ,$
FIELD=DEDUCTIBLE          ,ALIAS=DEDUCTIBLE           ,A3    ,A3    ,$
FIELD=GARAGE_LOCATION     ,ALIAS='GARAGING LOCATION'  ,A12   ,A12   ,$
FIELD=TRANS               ,ALIAS=TRANS                ,A2    ,A2    ,$
FIELD=USAGE               ,ALIAS=USAGE                ,A12   ,A12   ,$
FIELD=VEHICLE_USAGE_CLASS ,ALIAS='VEHICLE USE CLASS'  ,A2    ,A2    ,$
FIELD=VEHICLE_RATING      ,ALIAS='VEHICLE RATING'     ,A1    ,A1    ,$
FIELD=OWNER_POLICY        ,ALIAS='OWNER POLICY'       ,A6    ,A6    ,$
FIELD=PRINCIPLE_DRIVER    ,ALIAS='PRINCIPLE DRIVER'   ,A12   ,A12   ,$

SEGNAME=XOTHDR,  PARENT=VEHICLE, OCCURS=VARIABLE,$
FIELD=OTHER_DRIVERS       ,ALIAS='OTHER DRIVER'       ,A6    ,A6    ,$
FIELD=DRVCNT              ,ALIAS=ORDER                ,I4    ,I4    ,$
```

The following sections summarize the syntax for each type of declaration and provide detailed explanations for each attribute.

Reference: Access File Attributes

The following chart lists Access File attributes generated during synonym creation. Note that attributes vary depending on whether the synonym is generated based on a user requirements table or a connection string.

Keyword	Description
ALIAS	Full Model 204 field name, if its length exceeds 66 bytes.
TYPE	Model 204 key field type.

Keyword	Description
ATTRIBUTE	ATTRIBUTE will incorporate Model 204 FOR EACH VALUE processing on a BY field in a server COUNT request. The field defined with ATTRIBUTE=FRV must be a Model 204 Ordered field or a Model 204 KEY/FRV field.
ACCDATA	ACCDATA will instruct the adapter to generate either character string or numeric HLI selection tests for Model 204 fields that do not have a Model 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Syntax: How to Enter an Account Declaration

The account declaration indicates authorization to access Model 204 files. At most, one account declaration is required; if included, it must be the first declaration in the Access File

```
[[ACCOUNT=userid[account], ACCOUNTPASS=password,] IFAMCHNL=channel ,$]
```

where:

userid

Is the user ID of 1-10 characters by which you are known to Model 204.

account

Is a string of 1-10 characters to be used only if an account under which the user will log in to Model 204 must be supplied.

password

Is the 1- to 16-character password for the account.

channel

Is the 1- to 8-character site specific CRAM channel name. The default is IFAMPROD for z/OS.

Reference: ACCOUNT and ACCOUNTPASS Attributes

The ACCOUNT and ACCOUNTPASS attributes describe Model 204 accounts, user IDs, and passwords. They protect Model 204 files from unauthorized access. The account must have authorization to read the Model 204 file or group.

You can omit the ACCOUNT and ACCOUNTPASS attributes from the declaration if you:

- Use the attribute CONNECTION in the segment declaration of the Access File.
- Specify the values with the adapter M204IN SET M204ACCNT and M204PASS commands. These SET commands also enable you to change ACCOUNT and ACCOUNTPASS values during the session. For information about adapter SET commands, see *Customizing the Model 204 Environment* on page 1619.

Reference: IFAMCHNL Attribute

The IFAMCHNL attribute describes the CRAM channel name for z/OS. You must specify the CRAM channel if more than one ONLINE or IFAM2 job is running or if the CRAM channel name is not IFAMPROD. For the channel name used at your site, check with your Model 204 database administrator or the IFAM2 systems support staff .

If the attribute CONNECTION is provided in the segment declaration, the channel name is taken from the appropriate SET CONNECTION_ATTRIBUTES command.

Syntax: How to Enter a Segment Declaration

The segment declaration in the Access File establishes the link between the server Master File and the Model 204 file or group. Attributes that constitute the segment declaration are: SEGNAME, FILE, PASS, CONNECTION, KEYFLD, IXFLD, RECTYPE, RECTVAL, and ACCESS. Values for SEGNAME, FILE, and PASS are required. The attributes KEYFLD and IXFLD apply to multi-segment structures. The ACCESS attribute applies to OCCURS segments

```
SEGNAME=name, FILE=m204file, PASS=password [,CONNECTION=connection_name]
  [,KEYFLD=pfield] [,IXFLD=cfield] [,RECTYPE=rtfield] [,RECTVAL=rtvalue]
  ,ACCESS={ALL|nnnn[/xxx]|AUTO[/xxx]}
```

where:

name

Is the SEGNAME value from the Master File.

m204file

Is the 1- to 8-character name for the Model 204 file.

password

Is the 1- to 8-character read password for the Model 204 file. The value can be blank if a password does not exist.

connection_name

Is the valid connection that was previously set by the SET CONNECTION_ATTRIBUTES command.

pfield

Is the field name of the common field from the parent segment, and is used to implement the Join relationship. It is required in dependent segment declarations for multi-segment structures.

cfield

Is the field name of the common field from the dependent segment, and is used to implement the Join relationship. It is required in dependent segment declarations for multi-segment structures.

rtfield

Is the field name for the Model 204 record type field that exists when a Model 204 file has several logical record types.

rtvalue

Is the value that identifies the Model 204 record type or a field name.

ALL

nnnn/xxx

AUTO/*xxx*

Indicates how OCCURS segments should be processed when all occurrences cannot fit into the buffer at once and a (FOC4883) IFMORE FAILED message results. ALL is the default value. These options are described in *Access Files for Model 204* on page 1607.

Note:

- For multi-segment structures, the order of the Access File segment declarations must correspond to the order in the Master File.
- Dummy segments used to link unrelated segments do not have corresponding segment declarations in the Access File.

Reference: SEGNAME Attribute

The SEGNAME value must be the first attribute in each Access File segment declaration. It must be identical to the SEGNAME value in the Master File.

Reference: FILE and PASS Attributes

The FILE and PASS attributes are required in each Access File segment declaration. They describe Model 204 file security information.

The value for the FILE attribute is the name of the Model 204 file or group that contains the logical record type. The value for the PASS attribute is either the read password associated with the Model 204 file or blank if the password does not exist. For both attributes, acceptable values are 1- to 8-character values.

Access Files can be encrypted to prevent access to this security information.

Reference: CONNECTION Attribute

The CONNECTION attribute may be used in the Access File on the segment level. The value of this attribute is the valid connection_name that was previously set by the SET CONNECTION_ATTRIBUTES command. If this attribute is used in the Access File, the file-level parameters must be omitted or they will be overwritten by the value from the SET command.

```
SEGNAM=RNG, CONNECTION=M204, FILE=RNG, $
  FIELD=FKEY, TYPE=KEY, ATTRIBUTE=FRV, $
  FIELD=FORC, TYPE=KEY, $
  FIELD=FORN, TYPE=KEY, $
  FIELD=FRNG, TYPE=RNG, $
```

Syntax: How to Declare KEYFLD and IXFLD

The KEYFLD and IXFLD attributes identify the common fields for parent/child relationships in a multi-segment Master File. These relationships are referred to as Embedded Joins, server views, or cross-references.

For each dependent segment, the KEYFLD and IXFLD attributes identify the field names of the common or shared field that implements the Embedded Join. The parent field supplies the value for cross-referencing; the dependent field contains the corresponding value. The adapter implements the relationship by matching values at run time.

The value for the KEYFLD attribute is a 1- to 66-character field or alias name from the parent segment. The value for the IXFLD attribute is a 1- to 66-character field or alias name from the dependent segment.

Note:

- Include the pair of attributes in Access File segment declarations for dependent segments. Do not specify them in the segment declaration for the root segment.
- Do not specify KEYFLD and IXFLD attributes for dependent segments of a virtual parent segment (PARENT=DUMMY).

A JOIN can be based on more than one field in the host and cross-referenced logical record types. If the Model 204 file uses multiple fields to establish a relationship or link between logical record types, you can specify concatenated fields in an Embedded Join (you can also specify multiple fields with the Dynamic JOIN command).

In a multi-field Embedded Join, the KEYFLD and IXFLD values consist of a list of the component fields separated by slashes (/). Additional Access File attributes are not required.

```
KEYFLD = field1/field2/....
IXFLD = cfield1/cfield2/....
```

where:

fieldl/...

Is a composite of up to 16 key fields from the parent segment. Slashes are required.

cfieldl/...

Is a composite of up to 16 key fields in the dependent segment.

The adapter compares each field pair for comparable data formats prior to format conversion. It evaluates each field pair with the following rules:

- Any of the Model 204 key fields listed in Field Declarations can be used to create an Embedded Join.
- Parent and dependent fields must be real fields.
- All participating fields for either the parent or dependent file must reside in the same segment.
- KEYFLD and IXFLD attributes must specify the same number of fields. If the format of the parent field is longer than the format of the dependent field, its values are truncated. If the format of the parent field is shorter, its values are padded with zeros or blanks.
- The KEYFLD and IXFLD attributes can consist of a maximum of 16 concatenated fields in high to low significance order. You must specify the field order; the order determines how the values will be matched.
- Each pair of fields in the KEYFLD/IXFLD attribute must have comparable data formats. The formats of the parent fields must be compatible with the formats of the dependent fields as specified in the Model 204 file description.

To implement Joins, the Model 204 DBMS converts the alphanumeric server field formats to equivalent Model 204 field formats in order to perform the necessary search and match operations. When the Model 204 DBMS returns the answer set of matched values, it also converts the values back to alphanumeric formats.

Internally, each KEYFLD value is passed to the Model 204 DBMS in the IFFIND call to retrieve all matching records (segment instances) for the dependent segment.

Syntax: How to Declare RECTYPE and RECTVAL

The RECTYPE and RECTVAL attributes identify the Model 204 record type field and its corresponding value. A Model 204 record type field exists in a Model 204 file description when the file has more than one logical record type. If the Model 204 file consists of one logical record type, it may contain an optional record type field.

The value for the RECTYPE attribute is the 1- to 66-character field or alias name of the Model 204 record type field. The value for the RECTVAL attribute is the identifying value for the record type field; it can be up to 255 characters in length or a field name. The RECTVAL attribute is required whenever the RECTYPE attribute is used.

The RECTYPE and RECTVAL attributes are required in each Access File segment declaration (including the one for the root) if the Model 204 file contains record types.

Note: Do not use the name RECTYPE as a field name in the Master File.

ACCESS

OCCURS segments are generally described only in the Master File and do not require corresponding segment declarations in the Access File. However, sometimes the number of occurrences retrieved for an OCCURS segment exceeds the Model 204 buffer capacity, causing an error condition. You can alter the processing of OCCURS segments by including a segment declaration in the Access File.

The Model 204 buffer is used to return data to IFAM2 applications. Buffer capacity is sometimes exceeded when both of the following conditions exist:

- A multiply occurring field repeats a large number of times (for example, 2000 occurrences).
- The total number of occurrences multiplied by the size of the OCCURS segment is greater than the size of the IFAMBS buffer (the IFAMBS size is specified as a parameter in the Model 204 startup procedure).

The size of the IFAMBS buffer is determined by a predefined formula:

```
IFAMBS := (LIBUFF * 7) + 284 s 32688
```

When the buffer capacity is exceeded, the Model 204 DBMS sends the adapter the following message:

```
IFMORE error M204.0903 - Results too long
```

This causes the server message:

```
(FOC4883) IFMORE FAILED
```

If you encounter this condition, you can include a segment declaration in the Access File to alter the number of occurrences retrieved. The adapter incorporates your specification in its HLI call to the DBMS.

An OCCURS segment declaration in an Access File consists of two attributes, SEGNAME and ACCESS. The ACCESS attribute indicates the type of processing to be performed.

```
SEGNAME=name, ACCESS={ALL|nnnn[/xxx]|AUTO[/xxx]} ,$
```

where:

name

Is the SEGNAME value for the OCCURS segment from the Master File.

ALL

Indicates that the adapter will attempt to retrieve all occurrences of the OCCURS segment. ALL is the default value.

Note: If the ACCESS attribute is omitted, the ALL setting is assumed.

nnnn

Specify the largest number of occurrences that exists for the multiply occurring field. Based on the nnnn value, the adapter may manipulate the length of the QTBL buffer for the user session and then reset it to its original length.

AUTO

Indicates that the adapter will retrieve up to 5000 occurrences. If there are more than 5000 occurrences, use nnnn instead of AUTO.

/xxx

Instructs the adapter to issue multiple IFMORE calls, with each call retrieving xxx occurrences. This parameter is optional.

Append to the nnnn or AUTO parameters to indicate that you want to control the number of records to be retrieved by each IFMORE call. If xxx is larger than the maximum allowed (based on the size of the buffers used by the Adapter for Model 204), you will receive the message:

```
(FOC4873) SPECIFIED NUMBER OF INSTANCES CANNOT FIT INTO BUFFER
```

To correct the problem, decrease the xxx value and rerun the request.

Note:

- You can turn this feature on or off by defining the OCCURS segment in the Access File. OCCURS segments not described in the Access File undergo normal processing, which means that the ALL setting is assumed and all occurrences are retrieved at one time.
- If the SEGNAME attribute is specified in the Access File without the ACCESS attribute, ACCESS=ALL is assumed by default.
- Use this feature when the following message occurs:

  ```
  IFMORE error M204.0903 Results too long
  ```

Syntax: How to Enter Field Declarations

Field declarations in the Access File provide an alternate method for describing certain Model 204 field characteristics. Include a field declaration if the Model 204 field has a Model 204 key designation or if the Model 204 field name (the ALIAS attribute) exceeds 66 characters.

Reference: Supporting Types of Key Fields

The Model 204 DBMS supports a variety of key field types. The adapter supports the Model 204 key designations with comparable abbreviations called suffix operators. You describe these operators with the TYPE attribute in the Access File. The actual Model 204 key designation determines the proper suffix operator to apply.

The following chart lists Model 204 key designations and the corresponding suffix operators.

Model 204 Key Designation	Master File Suffix Operator
Key	KEY
Key, invisible	IVK
Ordered Character	ORA
Ordered Numeric	ORN
Numeric Range	RNG
Numeric Range, invisible	IVR

Model 204 Key Designation	Master File Suffix Operator
Ordered Character, invisible	IOA
Ordered Numeric, invisible	ION

Note: Master Files created in earlier server releases may contain field declarations with suffix operators for Model 204 key fields. While this earlier method is supported, the recommended method is to specify suffix operators in the Access File.

Syntax: How to Declare Field Attributes in an Access File

```
FIELD=field [,ALIAS=m204field] [,TYPE=suffix] [,ATTRIBUTE=FRV],
  [ACCDATA= {STR|NUM}] ,$
```

where:

m204field

Is a Model 204 field name that exceeds 66 characters. 256 characters is the maximum amount.

Note: Omit the ALIAS attribute if the ALIAS attribute in the Master File already specifies the Model 204 field name.

suffix

Is one of the suffix operators KEY, IVK, ORA, ORN, RNG, IVR, IOA, ION.

FRV

This attribute value incorporates Model 204 FOR EACH VALUE processing on a BY field in a server COUNT request.

STR

This attribute value will generate character string HLI selection tests for Model 204 fields that do not have a Model 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

NUM

This attribute value will generate numeric HLI selection tests for Model 204 fields that do not have a Model 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Two additional field attributes, ATTRIBUTE and ACCDATA are available:

- ATTRIBUTE will incorporate Model 204 FOR EACH VALUE processing on a BY field in a server COUNT request. The field defined with ATTRIBUTE=FRV must be a Model 204 Ordered field or a Model 204 KEY/FRV field.
- ACCDATA will instruct the adapter to generate either character string or numeric HLI selection tests for Model 204 fields that do not have a Model 204 key specification (Non-Ordered, Non-KEY, Non-Numeric Range).

Customizing the Model 204 Environment

Each Master File for a Model 204 file must have a corresponding Access File. The name of the Access File must be the same as that of the Master File. In z/OS, the Access File PDS is allocated to ddname ACCESS in the server JCL. The Access File associates a segment in the Master File with the Model 204 logical record type it describes.

The Adapter for Model 204 provides several parameters for customizing the environment and optimizing performance. These parameters control or identify Model 204 account security, the size of all adapter buffers, the size of the Model 204 FTBL buffer, null values in reports, locked records, and thread management. Information Builders strongly recommends that you set these parameters when you configure the Adapter for Model 204 (see *Connection Attributes for Model 204* on page 1582).

However, you can also use the legacy adapter SET commands to change parameters that govern the adapter's behavior. These parameters control or identify Model 204 account security, the size of all adapter buffers, the size of the Model 204 FTBL buffer, null values in reports, locked records, and thread management. To display current adapter parameter settings, issue the adapter M204IN SET ? query command. You can issue these commands from an RPC. They remain in effect for the duration of the server client task or until you change them.

Note: If the attribute CONNECTION is provided in the Access File, then values of parameters that were placed in the appropriate SET CONNECTION_ATTRIBUTES command will be used.

Syntax: How to Set Adapter Parameters

The syntax for adapter SET commands includes the ENGINE environment qualifier and the M204IN keyword

```
ENGINE M204IN SET command value
```

where:

command

Is an Adapter for Model 204 command.

value

Is an acceptable value for the adapter command.

Specifying a User Account and Password

You can supply Model 204 user IDs, accounts, and passwords with the adapter M204IN SET M204ACCNT and M204PASS commands. Authorized accounts, user IDs, and passwords protect read access to Model 204 files or groups of files.

The M204IN SET M204ACCNT and M204PASS commands provide an alternative to supplying security information in encrypted Access Files. You can also use the M204IN SET M204ACCNT and M204PASS commands to override existing account and password values specified with the ACCOUNT and ACCOUNTPASS attributes in the Access File.

Syntax: How to Specify a User Account and Password

To specify an account, issue the following from the command level

```
ENGINE M204IN SET M204ACCNT userid [account]
```

where:

userid

Is the 1-10 character user ID by which you are known to Model 204.

account

Is a string of 1-10 characters, to be used only if an account under which the user will log in to Model 204 must be supplied.

To specify a password for an account, issue the following from the command level

```
ENGINE M204IN SET M204PASS password
```

where:

password

Is a 1- to 16-character password for the account.

Note: If you do not issue the SET commands or you leave the values blank, they default to the values of the ACCOUNT and ACCOUNTPASS attributes specified in the Access File.

Specifying the Capacity of Adapter Buffers

You can set or change the maximum size of all adapter buffers with the M204IN SET MAXMBUFF command.

Before you issue the M204IN SET MAXMBUFF command, consider that the MAXMBUFF value:

- Should be greater than the larger of two Model 204 buffer settings, LIBUFF and LOBUFF. To identify which buffer is larger, check the parameters in the Model 204 start-up procedure or use the adapter Trace Facility (SET TRACEON=M204IN) to display the settings.
- Should be equal to the size of the largest segment in the Master File if the segment size is greater than the Model 204 LIBUFF and LOBUFF buffer settings. Having the maximum buffer size equal to that of the largest segment prevents S0C4 ABEND conditions.

Syntax: How to Specify the Capacity of Adapter Buffers

```
ENGINE M204IN SET MAXMBUFF nnnn
```

where:

nnnn

Is the maximum size in bytes. For example, specify 4096 for a MAXMBUFF value of 4K. Unless you explicitly issue this command, it is not used and there is no default value.

Controlling the Size of the FBTL Buffer

The M204IN SET FTBL command enables you to control the size of the Model 204 FTBL buffer. The LFTBL parameter, specified in the Model 204 start-up procedure, governs the size of the FTBL buffer. You can change it for the current client session only. Issue this command when a Model 204 message indicates that the LFTBL value is too small.

Syntax: How to Control the Size of the FTBL Buffer

```
ENGINE M204IN SET FTBL nnnn
```

where:

nnnn

Is the FTBL size in bytes. For example, specify 4096 for an FTBL value of 4K.

You should set FTBL to a value larger than the previously defined Model 204 LFTBL value.

Note: The FTBL value you set applies for the current user session only; it is reset when the session is ended.

Indicating Missing Data on Reports

With the adapter M204IN SET MISSING command, you can control the display of Model 204 null data on reports. Null data is translated into the server missing data display value. The default NODATA display value is the period (.). When the adapter MISSING attribute is set to ON, the edit specification for all IFGET calls includes an (L) format code. If Model 204 returns a zero in the first byte, the adapter considers that field to be missing.

Note: IFFIND calls contain IS PRESENT or IS NOT PRESENT criteria only when the request includes a server IF or WHERE MISSING selection test.

Syntax: How to Indicate Missing Data on Reports

```
ENGINE M204IN SET MISSING {ON|OFF}
```

where:

`ON`

Uses the NODATA value to display Model 204 null data.

`OFF`

Null values are not represented on reports. OFF is the default value.

Note:

- The default setting can affect the results of server DML SUM or COUNT aggregate operations. Null values may be counted or averaged in as existing values.
- You must issue the M204IN SET MISSING ON command in order to specify screening conditions (IF or WHERE MISSING tests) for null values.

Locking Records to Ensure Accurate Data Reports

When retrieving records, the Model 204 DBMS usually holds locks on the appropriate records to ensure accurate data for reports. The locks prevent other users from updating the target records while Model 204 constructs answer sets. The adapter generates standard IFFIND calls that lock the found set in SHR mode.

For certain circumstances, you can issue the adapter SET READWOL (read without locks) command to instruct the adapter to issue an IFFWOL (find without locks) HLI call. Use the following syntax to turn the setting ON:

```
ENGINE M204IN SET READWOL ON
```

For information about the design and performance considerations involved in deciding when to use the Model 204 IFFWOL command instead of the IFFIND command, consult the *Model 204 Host Language Adapter Programming Guide*.

Controlling Thread Management

The adapter M204IN SET SINGLETHREAD command enables you to control Model 204 thread management during record retrieval. A thread is a communications path or link between the adapter and the Model 204 DBMS. Adapter requests (generated HLI calls) are transmitted using threads. Threads are de-allocated after each adapter request is processed.

There are two types (or modes) of thread management:

- **Multiple.** The adapter creates as many threads as it needs whenever they are needed to access files.
- **Single.** The adapter creates one thread for all file access. This restricts processing to one request.

Syntax: How to Change the Mode of Thread Management

To change the mode, issue the following from the command level

```
ENGINE M204IN SET SINGLETHREAD {ON|OFF}
```

where:

`ON`

Forces the adapter to use a single thread for all file access.

`OFF`

Allows the adapter to use multiple threads. OFF is the default value.

For example, if a request joins 10 files and the SINGLETHREAD setting is ON, only one thread is used to access all of the files. If the setting is OFF, up to 10 threads may be used. For performance reasons, the OFF setting is recommended.

Displaying Adapter Defaults and Current Settings

The adapter M204IN SET ? query command displays adapter defaults and current settings. Issue the following:

```
ENGINE M204IN SET ?
```

Using Customized Security Exits

Your database administrator can provide Model 204 logon and account information with security exits written in COBOL or Assembler. The adapter supports this method of supplying security information through exit parameters.

The security exit is site specific and must be named M204EXT. The M204EXT program is executed only when the ACCOUNT name and PASSWORD have not been previously defined to the adapter either in the Access File, with a Dialogue Manager &variable, or by issuing the M204IN SET commands.

Syntax: How to Specify a Security Exit

The adapter calls the M204EXT exit with the following syntax

```
CALL M204EXT(USR,ACCNT,ACCNTP)
```

where:

`USR`

Is the 8 character TSO or MSO logon userid that is passed to the security exit.

`ACCNT`

Is the 16 character Model 204 account (userid) that is returned from the exit.

`ACCNTP`

Is the associated 16 character account password that is returned from the exit.

Reference: Security Exit Processing

When a user issues a report request to access Model 204 data, the adapter calls the M204EXT exit using standard IBM calling conventions and passes the three parameters to the exit. Based on the supplied userid (USR), the exit returns the account and password values to the adapter, which uses the returned values to generate the IFSTRTN call. (The IFSTRTN call initiates an internal logon to the Model 204 DBMS before the user request for services is executed.) After generating the IFSTRTN call, the adapter erases the values from memory.

To use the security exit, the database administrator must:

1. Write the security exit in COBOL or Assembler. The program name must be M204EXT.

   A sample Assembler M204EXT security exit program is provided in the 'FOCM204.DATA(M204EXT)' data set. The sample can be modified according to your site standards.

2. Link edit the object module into a load library with AMODE(31). For example

```
//LINK     EXEC PGM=IEWL,
//         PARM='LET,NCAL,SIZE=(1024),LIST'
//OBJLIB   DD   DSN=objlib.LOAD,DISP=SHR
//SYSLMOD  DD   DSN=loadlib.LOAD,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(10,1))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE OBJLIB(M204EXT)
  MODE  RMODE(31),AMODE(31)
  ENTRY M204EXT
  NAME M204EXT(R)
/*
```

   where:

   `objlib.LOAD`

   Is the name of the partitioned data set (PDS) that contains the M204EXT object code.

   `loadlib.LOAD`

   Is the load library that is to contain the M204EXT load module.

3. Concatenate the load library (allocated to ddname SYSLMOD in Step 2) to ddname USERLIB in the TSO CLIST that you use to invoke the server, or to ddname STEPLIB in your TSO logon CLIST.

   For MSO, either link the M204EXT load module into the 'prefix.FOCLIB.LOAD' data set, or allocate it to ddname STEPLIB.

Adapter Tracing for Model 204

The Adapter for Model 204 Tracing Facility is activated from the Web Console main page.

Procedure: How to Activate Adapter Tracing

1. From the Web Console menu bar, select *Workspace*, then *Diagnostics* and *Traces*. The Trace pane opens.
2. Click *Enable Traces*.

The default traces include the Adapter for Model 204 tracing information.

Chapter 65

Using the Adapter for MongoDB

The adapter for MongoDB allows applications to access MongoDB data sources.

In this chapter:

Introducing the Adapter for MongoDB

MongoDB is an open source NoSQL database. This is a document-oriented database designed to store extremely large amounts of data.

MongoDB stores documents consisting of name/value pairs, arrays, and other structures found in JSON (JavaScript Object Notation). Documents are organized in collections. Collections and documents are similar to tables and rows in a relational database.

NoSQL is sometimes defined as *Not Only SQL*, but MongoDB does not provide a SQL client or API. Unlike relational databases, MongoDB does not use pre-defined schemas.

The MongoDB Connector for BI is the JDBC driver available with MongoDB Enterprise Edition. This driver acts as a translation layer between the database and the reporting tool.

Preparing the MongoDB Environment

The following components are the prerequisites needed to use the adapter for MongoDB:

- **Java.** To use this Java-based adapter, you must have the JDK installed. MongoDB requires Version 1.8 or later. Java can be downloaded from *http://www.java.com*.

  The location of the JDK must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/java/jdk1.8.0_45
  ```

Alternatively, add this variable to the server environment. From the Data Management Console or the Web Console, go to the Workspace page. Expand the *Configuration Files* and *Miscellaneous* folders. Double-click *Environment - edaenv.cfg* and add JAVA_HOME to the file. For example

```
JAVA_HOME=/usr/java/jdk1.8.0_45
```

If you are using Windows, right-click *Computer* and click *Properties*. Then select *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

```
C:\Program Files\Java\jdk8\bin\server;C:\Program Files\Java\jdk8\bin;
```

- **BI Connector JDBC Driver for MongoDB.** The BI Connector JDBC driver is available from mongoDB website:

  ```
  mysql-connector-java-5.1.43/mysql-connector-java-5.1.43-bin.jar
  ```

Procedure: How to Configure the Java CLASSPATH

The location of the driver jar files must be specified to the server. You must extract the jar files to a location on your system and specify their location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console sidebar, click *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.

2. Expand the *Java Services* folder. Right-click *DEFAULT* and select *Properties*.

   The Java Services Configuration page opens.

3. Click the chevrons to expand Class Path.

   In the IBI_CLASSPATH box, enter the full location of the jar files shown below. The file names must be entered one per line.

   ```
   path/mysql-connector-java-5.1.43/mysql-connector-java-5.1.43-bin.jar
   ```

   Note: For a server running on Windows, use Windows syntax for directory names. For example:

   ```
   directory\ mysql-connector-java-5.1.43/mysql-connector-java-5.1.43-
   bin.jar
   ```

4. Scroll down and click the *Save and Restart Java Services* button.

Configuring the Adapter for MongoDB

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: MongoDB Adapter Configuration Settings

The Adapter for MongoDB is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the location of the MongoDB data source.

The following is the structure of the URL that is used to connect to your MongoDB server using the BI connector:

```
jdbc:mysql://host:port/database
```

For example:

```
jdbc:mysql://lnxx64r7:3309/test
```

where:

host

Is the DNS name or IP address of the system where the Mongo BI connector is running.

port

Is the port number on which the Mongo BI connector is listening.

database

Is the MongoDB database name you want to connect to

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Creating Synonyms With MongoDB

Synonyms define unique names (or aliases) for each MongoDB collection that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given MongoDB collection.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.
2. Under the MongoDB folder, select a connection and right click. Click Show DBMS objects for MongoDB (CON01).

   The Synonym Candidate dialog box opens.
3. From the *object type* drop-down list, select *Tables*.
4. Click *Next*.

   The list of tables opens.

5. Select the check box for the objects for which you want to create synonyms. If you want to change the name of the synonym from the default name, click the name and edit it as needed.
6. Optionally, expand the *Customize data type mapping*s section and enter a value for Longchar Length or numeric precision.
7. Click the purple arrow to create the synonym.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Chapter 66

Using the Adapter for MySQL

The Adapter for MySQL allows applications to access MySQL data sources. The adapter converts application requests into native MySQL statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the MySQL Environment

The Adapter for MySQL requires the MySQL Connector/J JDBC driver. For Release 5.x, Version 5.0.4 or higher is required. For Release 3.x, Version 3.1.14 or higher is required.

In order to report against Unicode data in MySQL, you must enable Unicode support in the reporting server. For more information, see *MySQL and Unicode* on page 1635.

Procedure: How to Prepare the MySQL Environment on Windows

1. Specify the location of the MySQL Connector/JDBC driver files in the CLASSPATH environment variable. You must specify the full location and name of the jar file. For example, if the jar file is located in C:\Program Files\MySQL\JDBC Driver, then specify:

   ```
   CLASSPATH= C:\Program Files\MySQL\JDBC Driver\mysql-connector-
   java-5.1.17-bin.jar
   ```

2. Specify the location of the Javarun time environment or development kit in the JAVA_HOME or JDK_HOME environment variable. Either is acceptable.

   You must specify the location where the run time environment is installed. You should see a sub-directory bin in that directory. For example, if you have the run time environment in C:\Program Files\java\jre6 then specify:

   ```
   JAVA_HOME=C:\Program Files\java\jre6
   ```

Alternately, if you have the full development kit installed in C:\Program Files\java \jdk1.6.0_17, then specify:

```
JDK_HOME= C:\Program Files\java\jdk1.6.0_17
```

3. Start (or restart) the server.

Note: You also need to restart the server if the driver has changed.

Procedure: How to Prepare the MySQL Environment on UNIX

Specify the location of several files:

1. Specify the location of the MySQL Connector/JDBC driver files in the $CLASSPATH environment variable.

   For example, if the files are located in /usr/driver_files, you would issue the following statements:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   To ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

   Alternatively, you could use the Web Console to specify the location:

   a. From the Workspace menu, select *Configuration/Monitor*.

   b. Expand the *Java Services* folder, right-click *DEFAULT*, and select *Properties*.

      The Java Services Configuration pane opens.

   c. Select the *Class path* section.

   d. Specify the full path to the MySQL Connector/J jar file in the IBI_CLASSPATH field.

   e. Click *Save and Restart Java Services*.

2. Specify the location of the Java Development Kit's installation directory in the $JDK_HOME environment variable.

   For example, if you want to set the location of the Java Development Kit to /usr/java, you would issue the following statements:

   ```
   JDK_HOME=/usr/java
   export JDK_HOME
   ```

3. Specify the location of the Java Virtual Machine's installation directory in the $LD_LIBRARY_PATH environment variable.

   For example, if you want to set the location of the JVM to /usr/j2sdk1.4.2_01/jre/lib/i386/server, you would issue the following statements:

```
LD_LIBRARY_PATH=/usr/j2sdk1.4.2_01/jre/lib/i386/server
export LD_LIBRARY_PATH
```

Note that if the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

4. Start (or restart) the server.

Note: You also need to restart the server if the driver has changed.

MySQL and Unicode

The Adapter for MySQL is implemented using JDBC. This implementation supports Unicode data stored in character fields with the CHARACTER SET set to UTF-8.

You must set the LANG environment variable in the edastart file or in a separate shell file before you start the server. For example, for American English you would export the following variable:

```
export LANG=EN_US.UTF-8
```

For details, see *Unicode Support* in the *Server Administration* manual.

Configuring the Adapter for MySQL

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

The SET CONNECTION_ATTRIBUTES command allows you to declare a connection to one MySQL database server and to supply authentication attributes necessary to connect to the server.

You can declare connections to more than one MySQL database server by issuing multiple SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query referencing that connection is issued (see *Overriding the Default Connection* on page 1640). You can include SET CONNECTION_ATTRIBUTES commands in an RPC or a server profile. The profile can be encrypted.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The *first* SET CONNECTION_ATTRIBUTES command sets the default MySQL database server to be used.

- If more than one SET CONNECTION_ATTRIBUTES command declares the same MySQL database server, the authentication information is taken from the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for MySQL

The MySQL adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Enter the location URL for the MySQL data source. The basic syntax is

```
jdbc:mysql://host/database
```

where:

host

Is the computer name or IP address on which the MySQL database is located.

database

Is the name of the database.

These are two examples:

```
jdbc:mysql://localhost/qatst
```

```
jdbc:mysql://edaaix52/qatst
```

Beginning with MySQL Release 4.1, you can reference additional MySQL connection properties in the URL. If you wish to do so, follow these guidelines: in the URL the first property must be preceded by the ? character, and the second and subsequent properties referenced in the URL must be preceded by the & character followed immediately by an | character, as illustrated in the following example.

Suppose that you wish to add the following connection properties:

```
sessionVariables=sql_mode=PIPES_AS_CONCAT
 zeroDateTimeBehavior=convertToNull
```

Enter the URL as follows:

```
jdbc:mysql://host/database?sessionVariables=sql_mode=P
IPES_AS_CONCAT&|
 zeroDateTimeBehavior=convertToNull
```

To use the adapter with SSL, you need to add the verifyServerCertificate, requireSSL, and useSSL properties, with useSSL last, as follows:

```
jdbc:mysql://host:port/server?verifyServerCertificate=false&|
requireSSL=true&|useSSL=true
```

Note: The URL must be entered as a single line, without a space after the | character.

Driver name

Name of the MySQL JDBC driver.

For Connector/J 8.0 and higher, use the following driver:

```
com.mysql.cj.jdbc.Driver
```

For Connector/J versions lower than 8.0, use the following driver:

```
com.mysql.jdbc.Driver
```

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

For explicit authentication:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES [connection]/userid,password
```

For password passthru authentication:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES connection/
```

where:

`MYSQL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

The name of the connection. You can give any name to the connection that you want.

`userid`

Is the primary authorization ID by which you are known to MySQL.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the MySQL database server named TEST with an explicit user ID and password:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES TEST/USERA,PWDA
```

The following SET CONNECTION_ATTRIBUTES command connects to the MySQL database server named TEST using password passthru authentication:

```
ENGINE MYSQL SET CONNECTION_ATTRIBUTES TEST/
```

Authenticating a User

There are two methods by which a user can be authenticated when connecting to a MySQL database server:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users.
- **Database or Password Passthru.** User ID and password received from the client application are passed to the MySQL database server for authentication.

When a client connects to the server, the user ID and password are passed to MySQL for authentication and are not authenticated by the server. To implement this type of authentication, start the server with security turned off. The server allows the client connection, and then stores an encrypted form of the client connection message to be used for connection to a MySQL database server at anytime during the lifetime of the server agent.

Overriding the Default Connection

Once all MySQL connections to be accessed have been declared using the SET CONNECTION_ATTRIBUTES command, there are two ways to select a specific MySQL connection from the list of declared connections:

- You can select a default connection using the SET DEFAULT_CONNECTION command. If you do not issue this command, the connection name value specified in the *first* SET CONNECTION_ATTRIBUTES command is used.
- You can include the CONNECTION= attribute in the Access File of the table specified in the current SQL query. When you include a connection name in the CREATE SYNONYM command from the Web Console, the CONNECTION= attribute is automatically included in the Access File. This attribute supersedes the default connection.

Syntax: How to Select a Connection to Access

```
ENGINE MYSQL SET DEFAULT_CONNECTION [connection]
```

where:

`MYSQL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name specified in a previously issued SET CONNECTION_ATTRIBUTES command. If omitted, then the local database server will be set as the default. If this connection name has not been previously declared, a FOC1671 message is issued.

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the last command will be the active connection name.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, a FOC1671 message is issued.

Example: Selecting a Connection to Access

The following SET DEFAULT_CONNECTION command selects the MySQL database server named TNSNAMEB as the default MySQL database server:

```
ENGINE MYSQL SET DEFAULT_CONNECTION datasource_name
```

Note: You must have previously issued a SET CONNECTION_ATTRIBUTES command for the datasource_name.

Controlling Connection Scope

This topic explains how to set the scope of logical units of work using adapters. This is accomplished by the SET AUTODISCONNECT command.

A connection occurs at the first interaction with the declared database server.

Syntax: How to Control the Connection Scope

```
ENGINE MYSQL SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`MYSQL`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing MySQL Metadata

This topic describes how to use CREATE SYNONYM for MySQL data sources. It also describes MySQL data type support.

Creating Synonyms

Synonyms define unique names (or aliases) for each MySQL table or view that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for MySQL

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Object name

Selecting this option adds the Object Name parameters to the screen.

Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *MySQL Data Type Support* on page 1648.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for MySQL synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=MYSQL ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=TEST,KEYS=1, WRITE=YES,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the MySQL tablename. The tablename may include owner (schema) name. For example, TABLENAME=[*owner.*]*tablename*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.

Keyword	Description
KEY	Specifies the columns that participate in the primary key without having to describe them first in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	Set to Yes to enable autoincrementing.
START	Initial value in incrementing sequence
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

MySQL Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Adapter for the MySQL Environment

This topic describes how to set the adapter for the MySQL environment.

PASSRECS

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Set PASSRECS

```
ENGINE MYSQL SET PASSRECS {ON|OFF}
```

where:

MYSQL

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE MYSQL SET ISOLATION {RU|RC|RR|SE}
```

where:

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

MySQL Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Chapter 67

Using the Adapter for NATURAL

The Adapter for NATURAL enables a client to execute a NATURAL program that receives and returns data via NATURAL work files. The adapter can be executed on USS, UNIX, and Windows platforms.

In this chapter:

Preparing the NATURAL Environment

In order to be able to access Adabas databases from Natural programs, you must set up the Adabas environment. For details, see *Using the Adapter for Adabas* on page 129.

You must then perform several operating system-specific tasks to set up the Natural environment itself.

Modifying Model Files

z/OS:

Model files with the jcl extension are supplied in the *home/etc* directory. They contain jobs to be executed on the mainframe. You must modify these file by supplying correct values in all lines marked with <– characters.

- **IWAYNATB.** Installs the supporting NATURAL program IWAYNATB in the Adabas NATURAL file that contains library SYSEXT. This program is used in synonym creation.

  You can change the library and/or program name. However, the new name(s) must be changed correspondingly in the natbdbs.acx file.

  If you choose a non-SYSEXT library, you must either place the SYSEXT library in the NATURAL profile steplib list or concatenate the new library with the SYSEXT library.

- **IWAYIVPB.** Installs the example NATURAL program IWAYIVPB and local data areas. IWAYTSTI/IWAYTSTO define input/output buffers processed by the program. You can install these NATURAL objects in any NATURAL library.

  You can change the program name, however, if you rename data areas, you must change the names correspondingly in the program IWAYIVPB source code. These objects are used to verify synonym creation.

Windows and UNIX:

Model files with the natural extension are supplied in the *home/etc* directory. These files must be processed by the SAG NATURAL utility SYSTRANS.

- **IWAYNATB.** You must install the Natural program in the library SYSEXT. This program is used in synonym creation.

  You can change the library and/or program name. However, the new name(s) must be changed correspondingly in the natbdbs.acx file.

  If you choose a non-SYSEXT library, you must either place the SYSEXT library in the NATURAL profile steplib list or concatenate the new library with the SYSEXT library.

- **IWAYIVPB.** You must install the Natural program along with the supporting data areas to any library (SYSTEM is default). IWAYTSTI/IWAYTSTO data areas define input/output buffers processed by the program.

  You can change the program name, however, if you rename data areas, you must change the names correspondingly in the program IWAYIVPB source code. These objects are used to verify synonym creation.

Setting Up the NATURAL Environment

On z/OS:

Prior to configuring the Adapter for NATURAL using the Web Console, you must edit the IRUNJCL that is used to initiate the server by allocating the following database and Natural load libraries to the server STEPLIB:

```
//IRUNJCL     PROC
//STEPLIB  DD DISP=SHR,DSN=EDABXV.SRV71.HOME.LOAD
//         DD DISP=SHR,DSN=SAGLIB.NAT314.LOAD
//         DD DISP=SHR,DSN=ADABAS.V742.LOAD
```

Alternatively, if the server is started from USS using edastart, export STEPLIB from the edastart file, as follows:

```
export STEPLIB=SAGLIB.NAT314.LOAD:ADABAS.V713.LOAD
```

On Windows:

Add Natural directory to the PATH.

On UNIX:

Execute the **sagenv** script as follows to set the environment:

```
. <SAG-root-directory>/sagenv
```

Either sagenv or sagenv.new is generated during Natural installation and setup. For details, refer to the SoftwareAG Natural documentation.

Maintaining the NATURAL Parameter File on Windows and UNIX

Use the SAG-provided utility, natparm, to maintain the NATURAL parameter file. It is good practice to create a copy of the standard system parameter file and use this copy to run the adapter. The name of the new file is stored as part of the connection information.

The parameter file must contain the following data channels and work file definitions:

Input Data Channels:

- value for CMOBJIN - %EDATEMP%\cmsynin
- value for CMSYNIN - %EDATEMP%\cmsynin
- value for CMPRINT - %EDATEMP%\cmprint

Work Files:

- value for CMWKF31 - %EDATEMP%\cmwkf31 (Type=SAG)
- value for CMWKF32 - %EDATEMP%\cmwkf32 (Type=SAG)

You must set the parameter ENDMSG ON to verify the Natural program execution.

Configuring the Adapter for NATURAL

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to NATURAL, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to NATURAL takes place when the first query that references that connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for NATURAL

The NATURAL Batch adapter is under the *Procedures* group folder

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Natural driver

Is the name of the NATURAL driver. This required value is installation dependent.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Is the password associated with the user name. This field is only displayed when the security mode of the server is non-trusted.

Natural Session Parameters

NATURAL session dynamic parameters (system specific). These parameters are required:

- FDIC is a system file containing cross-reference information and DDMs. If Predict is installed at your site, FDIC is the Predict dictionary file. If Predict is not installed at your site, FDIC is identical to either FNAT or FUSER and on the mainframe contains DDMs only. FDIC is specified with the FDIC profile parameter. FDIC includes two parameters with the values Database number and File number.
- INTENS. For related information, see *System-Specific Usage Notes* on page 1657.

See your Natural Reference documentation for details.

Adabas Session Parameters

On USS platforms, these parameters are usually provided in the DDCARD DD statement. They include four parameters: database number, device type, SVC number, and MODE. See the Software AG documentation for details.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to NATURAL, at connection time, for authentication.

```
ENGINE NATB SET CONNECTION_ATTRIBUTES [connection]/userid,password:
"parameters_list"
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to NATURAL CICS, at connection time, for authentication.

```
ENGINE NATB SET CONNECTION_ATTRIBUTES [connection]/: "parameters_list"
```

Trusted. The adapter connects to NATURAL as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE NATB SET CONNECTION_ATTRIBUTES [connection]/,: "parameters_list"
```

where:

NATB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. For details, see *Access File Attributes* on page 1667.

userid

Is the primary authorization ID by which you are known to the Adapter for NATURAL.

password

Is the password associated with the primary authorization ID.

parameters_list

The following parameters are supported (some optional and some required):

natnuc_name is the name of the NATURAL driver. This required parameter is NATURAL installation dependent.

natuser_id is the userid for NATURAL security. This parameter is optional.

natpwsd_pswd is the password for NATURAL security. This parameter is optional.

'*natparm_value*' is the NATURAL session dynamic parameters (system specific). These parameters are required. For details, see *System-Specific Usage Notes* on page 1657.

'*adaparm_value*' is applicable only on USS platforms, ADABAS session parameters, usually provided in the DDCARD DD statement, include four parameters: database#, device type, SVC#, and MODE. See the Software AG documentation for details.

Reference: System-Specific Usage Notes

- On z/OS, the NATPARM value contains parameters usually provided in the PARM parameter of the EXEC statement for the NATURAL Batch driver. For details, consult the NATURAL DBA. Providing the parameter INTENS=1 is highly recommended.
- On Windows, the NATPARM value contains dynamic parameters accepted by NATURAL in batch mode. For details, see the SAG documentation.

In addition, a list of parameters may provide the name of the NATURAL parameter file in the form PARM=filename. The PARM parameter is mandatory if the file name is different from *natparm*. The parameter file is created and maintained by the SAG provided program natparm, as described in *Preparing the NATURAL Environment* on page 1651.

- In UNIX, the NATPARM value contains the same information as on Windows, but, in addition, it may contain a buffer pool name in form BP=name. For information on Buffer pool usage, see the SAG documentation.

Example: **Sample Connection Declarations**

Windows:

```
ENGINE NATB SET CONNECTION_ATTRIBUTES CON3 /,: "NATNUC natural NATPARM
'PARM=IWAYPARM'"
```

UNIX:

```
ENGINE NATB SET CONNECTION_ATTRIBUTES CON3 /,: "NATNUC natural NATPARM
'PARM=IWAYPARM BP=NATBP'"
```

z/OS:

```
ENGINE NATB SET CONNECTION_ATTRIBUTES CON3 /,: "NATNUC NAT314BA NATPARM
'FDIC=(,21),INTENS=1' ADAPARM 'ADARUN DB=003,DE=3390,SVC=240,MODE=MULTI'"
```

Managing Metadata for NATURAL

When the adapter invokes a Natural program, it needs to know where to find the program and how to process its parameters. For each Natural program the adapter will access, you create a synonym that describes this information.

Creating Synonyms

A synonym defines a unique logical name (also known as an alias) for each Natural file, and one set of input/output parameters. The adapter requires that you generate a synonym for each Natural file you want to invoke with the adapter.

Synonyms define unique names (or aliases) for each transaction or procedure that is accessible from the server. Synonyms are useful because they hide the underlying transaction or procedure from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Creating a synonym generates a Master File and an Access File. These are metadata that describe to the server the Natural program name, parameters, and options.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for NATURAL

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Natural System/User file with program libraries:

Data base ID

The database ID.

File number

File password

If set, the password associated with the Natural system file.

Mask for library names

Natural library with Natural programs.

Enter a string for filtering library names, inserting the wildcard character (%) at the beginning and/or end of the string.

For example, enter ABC% to display libraries whose names begin with the letters ABC; %ABC to display libraries whose names end with the letters ABC; %ABC% to display libraries whose names contain the letters ABC at the beginning, middle, or end; or % to display all libraries.

Mask for program names

The mask for returning a list of program name from which to choose. Enter a string for filtering program names, inserting the wildcard character (%) at the beginning and/or end of the string.

Program names

Choose a program from the drop-down list.

No data area available

Check this box if your data is defined in either a local data area or a parameter data area. For more information, see the Software AG documentation on data area.

Note: If you check this box, you are re-prompted for database, file and password information on the next Create Synonym pane.

Leave this box unchecked if you are not using defined data.

Natural System/User file with data area definitions:

Data base ID

The database ID.

File number

A number that identifies the Natural System user file.

File password

If set, the password associated with the Natural system file.

Mask for the library names

The mask for returning a list of libraries from which to choose the one containing local/global data areas that describe the program parameters.

Mask for the Data Area names

Enter the mask for returning a list of the data areas from which you will choose the data area defining the program parameters.

Natural library

Select a library from the drop-down list.

Synonym name

The name of the synonym. You can retain the current name or change it.

Changing a synonym's name enables you to manage multiple synonym versions to reflect, for example, multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Input Data Area/Output Data Area

Specifies which data area's are for input parameters and which are for output parameters. You can specify the same data area as the source of both the input and output parameters.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Programming Requirements for NATURAL

To be invoked from the adapter, a NATURAL program must conform to the following rules:

- Program must read input parameters from the work file number 31.
- Program must write output data in the work file number 32.
- On USS platforms, the input parameter record is prefixed with the 2-byte length binary prefix containing the record length (excluding prefix itself).
- On USS platforms, each output record must be prefixed with the 2-byte length binary prefix containing the record length (excluding prefix itself). For example

  ```
  WRITE WORK FILE 32 VARIABLE #LEN MEM-LIST
  ```

  where:

  ```
  #LEN(I2)
  ```

  Contains the length of the structure MEM-LIST.

- On Windows and UNIX, the work files must be described in the Natural parameter file with type SAG. For related information, see *Preparing the NATURAL Environment* on page 1651.

Example: Creating a Synonym

The following example includes a NATURAL program and local data areas, with corresponding metadata.

```
/************************************************************************/
/* IWAYIVPB - Sample Natural Program for Natural Batch Adapter          */
/*                                                                      */
/* Input parameters for IWAYIVPB defined in LDA IWAYTSTI.               */
/* Output parameters for IWAYIVPB defined in LDA IWAYTSTO.              */
/*----------------------------------------------------------------------*/
DEFINE DATA
LOCAL USING IWAYTSTI /* INPUT PARMS FOR THE PROGRAM
LOCAL USING IWAYTSTO /* OUTPUT RECORD STRUCTURE
LOCAL

1 #LEN (I2) /*LENGTH OF DATA TO GET OR PUT
1 #MF (L) INIT <TRUE>
1 #OPSYS (A8)
1 REDEFINE #OPSYS
  2 #OP (A3)
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 FIRST-NAME
  2 NAME
  2 MAR-STAT
  2 SEX
  2 BIRTH
  2 DEPT
  2 JOB-TITLE
  2 CURR-CODE(1:5)
  2 SALARY(N9/1:5)
1 #ERROR-PARMS
  2 #NATPROG (A8)
  2 #NATMSG (A65)
  2 #NATERR (A7)
END-DEFINE

/*
MOVE *OPSYS TO #OPSYS
IF #OP NE 'MVS'
  #MF := FALSE
END-IF

/*
IF #MF
  READ WORK FILE 31 ONCE #LEN INPUT-PARMS
ELSE
  READ WORK FILE 31 ONCE INPUT-PARMS
END-IF

/*
IF #MF AND #LEN LT 16 /*REQUIRED FOR THIS PROGRAM*/
  MOVE ' INPUT PARAMETERS HAVE TO BE 16 BYTES' TO #NATMSG
  MOVE 80 TO #LEN
  WRITE WORK FILE 32 VARIABLE #LEN #ERROR-PARMS
  ESCAPE ROUTINE
END-IF
```

```
MOVE 145 TO #LEN
FIND ALL EMPLOY-VIEW WITH
 PERSONNEL-ID = PERSONNEL-ID-FROM THRU PERSONNEL-ID-TO
  IF NO RECORDS FOUND
    MOVE *PROGRAM TO #NATPROG
    MOVE ' REQUESTED EMPLOYEE NUMBERS NOT IN THE DATABASE' TO #NATMSG
    IF #MF
      MOVE 80 TO #LEN
      WRITE WORK FILE 32 VARIABLE #LEN #ERROR-PARMS
    ELSE
      WRITE WORK FILE 32 VARIABLE #ERROR-PARMS
    END-IF
    ESCAPE ROUTINE
  END-NOREC
  /* Create output record:
  MOVE BY NAME EMPLOY-VIEW TO OUTPUT-RECORD
  /* Send output record:
  IF #MF
    WRITE WORK FILE 32 VARIABLE #LEN OUTPUT-RECORD
  ELSE
    WRITE WORK FILE 32 VARIABLE OUTPUT-RECORD
  END-IF
END-FIND
END
```

NATURAL local data areas:

```
Local NATTSTI Library SYSTEM DBID 3 FNR 9
I T L Name                 F Leng Index/Init/EM/Name/Comment
    1 INPUT-PARMS
    2 PERSONNEL-ID-FROM    A 8
    2 PERSONNEL-ID-TO      A 8
Local NATTSTO Library SYSTEM DBID 3 FNR 9
I T L Name                 F Leng Index/Init/EM/Name/Comment
    1 OUTPUT-RECORD
    2 PERSONNEL-ID         A 8
    2 FIRST-NAME           A 20
    2 NAME                 A 20
    2 MAR-STAT             A 1
    2 SEX                  A 1
    2 BIRTH                D
    2 DEPT                 A 6
    2 JOB-TITLE            A 25
    2 CUR-CODE             A 3    (1:5)
    2 SALARY               N 9    (1:5)
```

To generate a synonym for the NATURAL program IWAYIVPB, enter the following information on the Create Synonym panes of the Web Console or the Data Management Console:

1. On the first Create Synonym pane, enter *3* in the Database ID field, enter *9* in the File number field, and enter the password associated with the Natural system file in the File password field. Enter *SYS%* as a mask to filter library names.

2. On the second pane, enter *IWAY%* to filter program names, select *SYSTEM* from the drop-down library list, and click *Next*.
3. On the third pane, choose *IWAYIVPB* from the drop-down list of programs. Leave *No data area available* unchecked since the data in this example is defined in a local data area. Click *Next*.
4. On the fourth pane, once again enter *3* in the Database ID field, enter *9* in the File number field, and enter the File password. Enter *SYS%* to filter library names, and click *Next*.
5. On the fifth pane, enter *IWAY%* to filter data area names, and choose *SYSTEM* from library the drop-down list. Click *Next*.
6. On the final Create Synonym pane, select *IWAYTSTI* from the list of Input Data Areas. Select *IWAYTSTO* from the list of Output Data Areas.
7. Click *Create Synonym*. The synonym is created and added under the specified application directory (baseapp is the default).

   A status window displays the message: *All Synonyms Created Successfully*
8. From the message window, click *Applications* on the menu bar.
9. Open the *baseapp* application folder in the navigation pane and click the synonym *IWAYIVPB*.
10. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File

```
FILENAME=IWAYIVPB, SUFFIX=NATB    , $
 SEGMENT=SEG1, SEGTYPE=S0, $
   GROUP=INPUT_PARMS, ALIAS=E1, USAGE=A16, ACTUAL=A16,
ACCESS_PROPERTY=(INTERNAL),$
    FIELDNAME=PERSONNEL_ID_FROM, ALIAS=E2, USAGE=A8, ACTUAL=A8,
ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=PERSONNEL_ID_TO, ALIAS=E3, USAGE=A8, ACTUAL=A8,
ACCESS_PROPERTY=(NEED_VALUE), $
 SEGMENT=SEG11, SEGTYPE=S0, PARENT=SEG1, $
   GROUP=OUTPUT_RECORD, ALIAS=E1, USAGE=A149, ACTUAL=A145, $
    FIELDNAME=PERSONNEL_ID, ALIAS=E2, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FIRST_NAME, ALIAS=E3, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=NAME, ALIAS=E4, USAGE=A20, ACTUAL=A20, $
    FIELDNAME=MAR_STAT, ALIAS=E5, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=SEX, ALIAS=E6, USAGE=A1, ACTUAL=A1, $
    FIELDNAME=BIRTH, ALIAS=E7, USAGE=P7, ACTUAL=P4, $
    FIELDNAME=DEPT, ALIAS=E8, USAGE=A6, ACTUAL=A6, $
    FIELDNAME=JOB_TITLE, ALIAS=E9, USAGE=A25, ACTUAL=A25, $
    FIELDNAME=CUR_CODE_POSN, ALIAS=E10, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=SALARY_POSN, ALIAS=E11, USAGE=A45, ACTUAL=A45, $
    DEFINE BIRTH_DAT/YYMD=BIRTH - 694324; $
 SEGMENT=SEG2, SEGTYPE=S0, PARENT=SEG11, OCCURS=5,
POSITION=CUR_CODE_POSN, $
    FIELDNAME=CUR_CODE, ALIAS=E12, USAGE=A3, ACTUAL=A3, $
 SEGMENT=SEG3, SEGTYPE=S0, PARENT=SEG11, OCCURS=5, POSITION=SALARY_POSN,
$
    FIELDNAME=SALARY, ALIAS=E13, USAGE=P10, ACTUAL=Z9, $
```

Generated Access File

```
SEGNAME=SEG1, CONNECTION=NAT1, TRANSACTION=SYSTEM:IWAYIVPB, $
```

Reference: Master File Attributes

The following Master File attributes describe Natural data segments:

Attribute	Description
FILENAME	The Master File name. This name may or may not match the stored procedure name.
SUFFIX	Identifies the adapter, and is always ADANAT.
SEGMENT	The segments in the description that are created when the synonym is generated. The segment names follow a logical format to provide uniqueness within the file.
FIELDNAME	The field name from the data area.

Attribute	Description
ALIAS	The real name of the field in the Natural file.
GROUP	The fields from the data areas that were redefined or that were defined as arrays.
USAGE	The display format and length of the field. This attribute determines how the value is displayed in reports. Values are determined based on the format and length specified by the ACTUAL attribute.
ACTUAL	The format and length of the field as described in the data area.

Reference: Access File Attributes

Attribute	Description
SEGNAME	Is the name of the input segment in the Master File.
CONNECTION	Indicates the connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. Defaults to the default connection.
TRANSACTION	Is the name of the library:program to be executed.

Invoking a NATURAL Program

To invoke a Natural program, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, or an SQL SELECT statement. (DML is the WebFOCUS internal retrieval language.) For information about the Data Manipulation Language, see the *Stored Procedure Reference* manual.

Syntax: How to Invoke a NATURAL Program Using TABLE

The syntax for invoking a Natural program using a TABLE command is

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the Natural program you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

IF/WHERE

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Example: Invoking the Natural Program NATTST Using TABLE

```
TABLE FILE NATTST
 PRINT PERSONNEL_ID NAME BIRTH SALARY
 IF PERSONNEL_ID_FROM EQ '11111111'
 IF PERSONNEL_ID_TO EQ '11211111'
 IF SALARY NE 0
END
```

Syntax: How to Invoke a NATURAL Program Using SELECT

The syntax for invoking a Natural program using a SELECT statement is

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the Natural program you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

WHERE

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Example: Invoking the Natural Program NATTST Using SQL SELECT

```
SQL
 SELECT PERSONNEL_ID, NAME, BIRTH, SALARY
  FROM NATTST
  WHERE PERSONNEL_ID_FROM = '11111111' AND
  PERSONNEL_ID_TO = '11211111' AND
  SALARY != 0;
END
```

Syntax: How to Invoke a NATURAL Program Using EX

```
EX [app_name_space/]synonym  1=parm1_val,..., N=parmN_val
EX [app_name_space/]synonym parm1_name=parm1_val,... ,
                            parmN_name=parmN_val
EX [app_name_space/]synonym [, parm1_val [,...[, parmN_val]]]
```

where:

app_name_space

Is the apps directory name under which the synonyms are stored. This value is optional.

synonym

Is the user-friendly name of a repository entry that represents the object to be executed in the vendor environment.

parm_name

Is the name of the parameter taken from the input metadata description.

1=parm1_val

N=parmN_val

parm1_name

parm1_val

parmN_val

Are the parameter values that match the input metadata description.

Note:

- Consecutive commas denote missing parameters in the positional form of the request.
- Values containing special characters (<equal sign> <space> <comma>) must be encapsulated in double or single quotation marks.

Example: Invoking the Natural Program NATTST Using EX

```
SET EXORDER=PGM/FEX
EX NATTST 11111111, 11211111
```

Chapter 68

Using the Adapter for NATURAL CICS Transactions

The Adapter for NATURAL CICS Transactions enables a client to execute a NATURAL program that receives and returns data using the Reporting Server API.

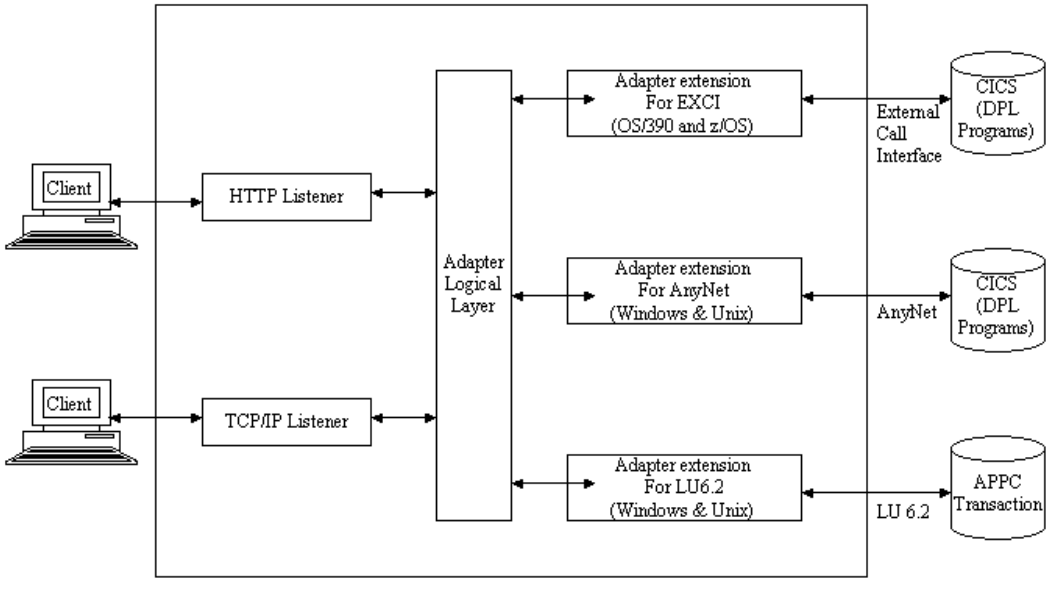
This adapter supports the following transport layers:

- EXCI for z/OS platforms.
- TCP62 through AnyNET for UNIX and Microsoft Windows platforms.
- TCP/IP for UNIX, Microsoft Windows, and z/OS platforms.
- LU6.2 for UNIX, Microsoft Windows, and z/OS platforms.

In this chapter:

Preparing the NATURAL CICS Environment

The following diagram shows the basic flow of a CICS transaction. It illustrates how the client application communicates with the adapter.



The communication to the NATURAL CICS Transactions region depends upon the platform the adapter is running on. If you are using UNIX or Microsoft Windows, you can use a communication transport of TCP 62 (AnyNET) for DPL programs or LU6.2 for APPC programs. If you are using z/OS, a direct NATURAL CICS Transactions External Call Interface is used for DPL programs (use of LU6.2 for APPC programs from z/OS is not supported in this release of the server).

To provide for transparent execution of NATURAL CICS Transactions programs, metadata describing the program input and output areas is held on the server. For DPL programs, the Natural Data Area describing DFHCOMMAREA is used as a source for metadata creation. For APPC-based programs, the Natural Data Area describing the storage layout for NATURAL CICS Transactions SEND and RECEIVE calls is used as a source for metadata creation: RECEIVE for input; SEND for output.

You can use the Web Console or the Data Management Console for configuration and to create/maintain the metadata associated with the programs to be executed. You can also use the Web Console to add or change adapter communication parameters.

NATURAL CICS Transactions Supported Platforms and Release Information

The following platforms are supported: Microsoft Windows, UNIX, and z/OS.

For Microsoft Windows and UNIX based servers that use AnyNET or LU6.2 to communicate with any NATURAL CICS Transactions region, the following minimum software release levels on a target are required:

- z/OS Version 1.1 or higher
- NATURAL CICS Version 4 or higher
- TCP/IP Direct Access, CICS TS2.3 or higher

Note: To use TCP62 or AnyNET, all VTAM options for AnyNET must be configured and AnyNET must be active.

For z/OS based servers, the following minimum software release levels are required:

- z/OS Version 1.1 or higher
- NATURAL CICS Version 4 or higher

Note: Because the EXCI option is being used, the adapter can only communicate with NATURAL CICS Transactions regions that are running on the same LPAR. Also, the STEPLIB of the EDASTART JCL member of the configuration data set needs to include the NATURAL CICS Transactions SDFHEXCI library.

NATURAL CICS and VTAM Configuration

These topics provide the setup and communications configuration information the adapter requires to communicate with NATURAL CICS Transactions.

After you configure any VTAM definitions, the major nodes they describe need to be started in the VTAM system. If you are using AnyNET, it should be tested from the Adapter for NATURAL CICS Transactions platform.

AnyNET VTAM Definitions

The AnyNET product requires three major VTAM nodes for its operation. The main node for AnyNET identifies the port. The following are examples of the three nodes. Replace relevant parameters with your site specific values.

Example: Setting Up the Major Node for the AnyNET Listener

```
**********  TCP62 MAJOR NODE DEFINITION FOR ANYNET
TCP62   VBUILD TYPE=TCP,DNSUFX=IBI.COM,PORT=397
TCP62G  GROUP  ISTATUS=ACTIVE
TCP62L  LINE   ISTATUS=ACTIVE
TCP62P  PU     ISTATUS=ACTIVE,NETID=MYNET
```

Note: Port 397 is used in this example. On UNIX, using a port number above 1000 is recommended due to the root privileges associated with using a port number under 1000. On z/OS, 397 is the default port; it is well established and frequently used.

Keep in mind, however, that the port you use must be the same on UNIX and z/OS. For example, if 1000 is specified on one platform, it must also be specified on the other.

Example: Setting Up the Major Node for Cross Domain Resources

If CDRDYN=YES is not specified in the ATCSTR00 VTAM startup parameter file, then the name of the machine that the Transaction Adapter is running on needs to be coded in the follow node:

```
        VBUILD TYPE=CDRSC
CDRSC62 NETWORK NETID=MYNET
CDRSC62 GROUP
EDABGR2 CDRSC ALSLIST=TCP62P
```

Note: EDABGR2 is the machine name on which the server is running. The machine name must be limited to eight bytes.

Example: Setting Up the Standard VTAM LU Definitions

```
        VBUILD TYPE=SWNET
MYNAMEPU  PU   ADDR=01,IDBLK=05D,IDNUM=10101,                X
               MAXPATH=3,                                    X
               PUTYPE=2,                                     X
               MODETAB=MTOS2EE,                              X
               DLOGMOD=PARALLEL,                             X
               ISTATUS=ACTIVE
MYLUNAME  LU   LOCADDR=2
```

LU6.2 VTAM Definitions

LU6.2 requires two major VTAM nodes for its operation. The main node for AnyNET identifies the port. The following are examples of these two nodes. Replace relevant parameters with your site specific values.

Example: Setting Up the Major Node for the AnyNET Listener

```
**********  TCP62 MAJOR NODE DEFINITION FOR ANYNET
TCP62   VBUILD TYPE=TCP,DNSUFX=IBI.COM,PORT=397
TCP62G  GROUP  ISTATUS=ACTIVE
TCP62L  LINE   ISTATUS=ACTIVE
TCP62P  PU     ISTATUS=ACTIVE,NETID=MYNET
```

Note: Port 397 is used in this example. On UNIX, using a port number above 1000 is recommended due to the root privileges associated with using a port number under 1000. On z/OS, 397 is the default port; it is well established and frequently used.

Keep in mind, however, that the port you use must be the same on UNIX and z/OS. For example, if 1000 is specified on one platform, it must also be specified on the other.

Example: Setting Up the Standard VTAM LU Definitions

```
        VBUILD TYPE=SWNET
MYNAMEPU  PU   ADDR=01,IDBLK=05D,IDNUM=10101,                X
               MAXPATH=3,                                    X
               PUTYPE=2,                                     X
               MODETAB=MTOS2EE,                              X
               DLOGMOD=PARALLEL,                             X
               ISTATUS=ACTIVE
MYLUNAME  LU   LOCADDR=2
```

CICS Connection and Sessions for Microsoft Windows and UNIX

For the Adapter for NATURAL CICS Transactions to be able to connect to NATURAL CICS Transactions (for both TCP62 and LU6.2 where supported), Connection and Session definitions are required. The definitions are dependent on the platform on which the adapter is running.

If you deploy the adapter on Microsoft Windows or UNIX, the connection and sessions definitions described in *Using the CEDA View Connection Command* on page 1676 and *Using the CEDA View Sessions Command* on page 1676 are required.

Note: For DPL program execution, all users must be able to access and run transaction CPMI (the mirror transaction).

Example: Using the CEDA View Connection Command

The following is an example of a CEDA view connection command that illustrates what is required:

```
CEDA  View Connection(  MYLU )
   Connection    :  MYLU
   Netname       :  MYLUNAME
   ACcessmethod  :  Vtam          Vtam | IRc | INdirect | Xm
   PRotocol      :  Appc          Appc | Lu61 | Exci
   SInglesess    :  No            No | Yes
   DAtastream    :  User          User | 3270 | SCs | STrfield | Lms
   INService     :  Yes           Yes | No
   ATtachsec     :  Verify        Local | Identify | Verify | Persistent
```

Example: Using the CEDA View Sessions Command

The following is an example of a CEDA view sessions command that illustrates what is required:

```
CEDA View Sessions(  MYLUNAME )
   Sessions      :  MYLUNAME
   Connection    :  MYLU
   MOdename      :  PARALLEL
   Protocol      :  Appc              Appc | Lu61 |
   MAximum       :  008 , 000         0-999
```

The MAximum parameter determines the number of concurrent sessions available to process requests. Its first value should be in the range of 4 - 255, and the second one should always be set to zero.

Note: Log mode must support parallel sessions.

Installing NATURAL Support Programs

Three model files with the jcl extension are supplied in the home/etc directory. You must modify these files and correct values supplied in all lines marked with <– character, and install them.

- **IWAYCICN.** This file contains the job to be executed on the mainframe to install a proxy transaction and the supporting run-time programs, AASNATC and AASSUBC, in CICS. You can change the Proxy transaction name. The new name will be used in the NATPROXY parameter when connection to CICS NATURAL is configured.

- **IWAYNATC.** This file contains the job to be executed on the mainframe to install the supporting NATURAL program IWAYNATC in the ADABAS NATURAL file containing library SYSEXT. These programs are used in metadata processing. You can change the library and/or program names. However, you must change the names correspondingly in the natcdbs.acx file. If you choose a non-SYSEXT library, you must either place the SYSEXT library in the NATURAL profile steplib list or concatenate the new library with the SYSEXT library.
- **IWAYIVPD.** This file contains the job to be executed on the mainframe to install the sample NATURAL program IWAYIVPD and the local data areas. IWAYTSTI and IWAYTSTO define input and output buffers processed by the program. You can install these NATURAL objects in any NATURAL library. You can change the program name. However, renaming of data areas requires corresponding changes in the program IWAYIVPD source code. These object names are used to verify synonym creation.

Configuring the Adapter for NATURAL CICS Transactions

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. In addition, for if you are using the TCP 62 or LU 6.2 communication protocols, you must configure a Listener node.

Configuring Communications Parameters for TCP 62 and LU 6.2

This procedure is required if you are using TCP 62 or LU 6.2 as your communications protocol.

Note that you can complete this step either before or after you declare connection attributes.

Procedure: How to Configure a Listener

Note: This step is required if you are using TCP 62 or LU 6.2 on Windows or UNIX.

1. From the Web Console menu bar, select *Workspace*, then *Configuration*.
2. On the expanded menu bar, select *New*, then *Listener*, and *CICST*. The Listener Configuration pane opens.
3. Type the appropriate values for the communications parameter.

   For descriptions of these parameters, see *Communications Parameters for CICS Transactions for TCP 62 or LU 6.2* on page 1678.

   Tip: You can also obtain details about each entry by clicking the *?* icon to the left of any parameter field on the communications configuration pane to access the Web Console Help.
4. Click *Save and Restart*.

Reference: Communications Parameters for CICS Transactions for TCP 62 or LU 6.2

This chart describes the parameters required to configure communications. To complete the configuration, click *Save and Restart Listener*.

Keyword	Description
NODE	8-character string Defines a logical name of a node block, which must be unique in the file. The logical name can be a maximum of 8 characters, must begin with a letter, and can include any characters, except semicolon and equal sign. A node block contains keyword-value pairs, which are enclosed in a BEGIN-END statement.
PARTNER_LU_NAME	string Defines the APPLID of the target CICS region.
LOCAL_LU_NAME	string Defines the LU name to be used. This value is taken from an LU entry in the major node. The value is also specified in the CICS connection/sessions definition.
MODE_NAME	string Defines the DLOGMODE parameter value.
CODEPAGE	string Defines the code page that the CICS region is using.
COMMUNICATION	Select one of the following options: ❑ TCP/IP for DPL program execution. This option displays additional parameters required for TCP 62. ❑ LU6.2 for APPC program execution. **Important:** Your selection in this field determines what additional configuration information that is required.
Additional required information for TCP/IP (This option displays additional parameters required for TCP 62.)	

Keyword	Description
VTAM_NETWORK_ID	string Defines the VTAM network ID. This value is taken from the AnyNET listener major node under parameter NETID.
CONNECT_LIMIT	number of seconds Defines the maximum time, in seconds, that the client waits for a connection response from the AnyNET environment. Provide a value greater than 0. (Do not enter a value of -1.)
PROXY	string Defines the name of the supplied proxy (user-written) program, which converts the Full-Function Server calling syntax to the syntax that is valid for the CICS program to be executed. This parameter is *optional*, and should be left blank for normal operations. Use it only at the direction of your local support personnel.
MIRROR	string Defines the IBM-supplied mirror transaction CPMI that the Adapter for CICS Transactions calls by default. If this transaction cannot be used, you can create another transaction name to point to the IBM-supplied mirror program DFHMIRS. Enter the alternate name in the MIRROR field.
NATPROG	string Identifies the NATURAL program to be executed. **Note:** This option does not apply to the Adapter for CICS Transactions.
HOST	string Defines the host that a client is connecting to or an IP address that a listener is listening on.

Keyword	Description
SERVER_ID	string Defines the machine name where the server is configured with the CICS AGENT NODE (Listener).
ANYNET_PORT	positive integer number Defines the port number that the AnyNET product is using. The default AnyNET port is 397, but any number can be used. For UNIX installations, a port number above 1000 is recommended.
Additional required information for LU6.2	
PROXY	string Defines the name of the supplied proxy (user-written) program, which converts the Full Function Server calling syntax to the syntax that is valid for the CICS program to be executed. This parameter is *optional*, and should be left blank for normal operations. Use it only at the direction of your local support personnel.
SIDE_INFO	string SIDE_INFO is a table with definitions for all partners to the currently active CICS system. CICS implements the side information table by means of the PARTNER resource. Partner resources are defined by the CEDA DEFINE command. To become known to an active CICS system, a defined partner resource must be installed using the CEDA INSTALL command.
HOST	string Defines the DNS name of the host LPAR of the target CICS region. You can also code the four part IP address.

Example: Sample Node Definitions Using LU6.2

```
NODE = CICSTLU6
BEGIN
  PROTOCOL = CICS
  CLASS = CICSCLIENT
  TARGET = CICS
  PARTNER_LU_NAME = EDBGM010
  LOCAL_LU_NAME = T29DPB41
  MODE_NAME = PARALLEL
  HOST = IBIMVS
  CODEPAGE = 037
  COMMUNICATION = LU62
END
```

Example: Sample Node Definitions Using TCP/IP (TCP 6.2)

```
NODE = LST_CICS
BEGIN
  PROTOCOL = CICS
  PORT = 8139
  CLASS = AGENT
END
NODE = SPGCSRV
BEGIN
  PROTOCOL = CICS
  PORT = 8139
  CLASS = AGENT
  TARGET = INTERNAL
  HOST = IBIMVS
  CODEPAGE = 037
  ANYNET_PORT = 397
END
NODE = CICSCLNT

BEGIN
  PROTOCOL = CICS
  CLASS = CLIENT
  TARGET = CICS
  INTEGER = HOBF
  FLOAT = IBM
  PARTNER_LU_NAME = EDBGM010
  LOCAL_LU_NAME = T29DPB04
  MODE_NAME = PARALLEL
  HOST = SPGCSRV
  CODEPAGE = 037
  COMMUNICATION = TCP
  VTAM_NETWORK_ID = USIBINET
  CONNECT_LIMIT = 60
  MIRROR = CPMI
  ATTACH_SECURITY = VERIFY
  SECURITY = C2B591CC2A92650028E8A570F3BE36F3
END
```

Declaring Connection Attributes

In order to connect to NATURAL CICS Transactions, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to NATURAL CICS Transactions takes place when the first query that references that connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

For detailed instructions about declaring connection attributes for your operating system, see *Configuring the Adapter on Microsoft Windows and UNIX* on page 1682 or *Configuring the Adapter on z/OS* on page 1686.

Configuring the Adapter on Microsoft Windows and UNIX

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. In addition, if you are using the TCP 6.2 or LU 6.2 communications protocol, you must configure a Listener node as described in *How to Configure a Listener* on page 1677. For TCP/IP, you only need to declare connection attributes.

Procedure: How to Declare Connection Attributes on Windows and UNIX

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Select a communications protocol: *TCP 62*, *LU 6.2*, or *TCP/IP* and click *Next*.

   For TCP/IP, the configuration parameters for the adapter are displayed.

   For TCP 62 and LU 6.2, you are prompted to either *Configure CICST listener* or to click *Next* to display the configuration parameters for the adapter. You can complete these tasks *in either order*, but both must be done to use the adapter.

   - If you click *Next*, the configuration parameters are displayed. However, in this scenario, you must then configure a Listener node, as described in *How to Configure a Listener* on page 1677.
   - If you click *Configure CICST listener*, the Web Console opens at the Special Services pane. For instructions, see *How to Configure a Listener* on page 1677.

     Once you have configured the Listener, return to the Add CICS Transaction for Natural to Configuration pane, where the Listener node is now available for selection.
7. Enter values for the parameters required by the adapter as described in the connection attributes reference.
8. Click *Configure*. The configured adapter is added to the Adapters list in the navigation pane.

Reference: Connection Attributes for NATURAL CICS Transactions on Windows and UNIX

The *CICS Transaction for Natural* adapter is under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Communication Protocol

Select one of the following: TCP/IP, TCP 62, LU 6.2. Subsequent options vary slightly depending on this selection.

Note: For TCP 62 and LU 6.2, you must *How to Configure a Listener* on page 1677 either before or after you declare connection attributes.

This selection is reflected in the Communication field on the second configuration pane.

Connection name

Is a logical name used to identify a specific set of connection attributes.

Node name

For TCP 62 or LU 6.2, a drop-down list of nodes is displayed if the Listener has already been configured. Select a node from the list.

Host name

For TCP/IP, enter the name of the host machine.

Application ID

For TCP 62 or LU 6.2, enter the application ID of the CICS Transactions region.

Mirror

Enter the name of the mirror transaction.

Port

For TCP/IP, enter the port number.

Proxy

Enter the name of a proxy program in NATURAL CICS. The default value is AASNATC. This value overwrites the PROXY parameter value provided when the Listener is configured.

Natural driver

Enter the name of the NATURAL nucleus in CICS. This value is NATURAL CICS-dependent.

Security

There are two methods by which a user can be authenticated when connecting:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the adapter, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the adapter, at connection time, for authentication.

User

Is the user name by which you are known to the adapter. This field is only displayed when the security mode of the server is non-trusted.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually on Windows and UNIX

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users:

```
ENGINE NATCICS SET CONNECTION_ATTRIBUTES [connection]
 [node_name]/userid,password: "parameters_list"
```

Password passthru. The user ID and password are explicitly specified for each connection and passed to CICS Transactions, at connection time, for authentication.

```
ENGINE NATCICS SET CONNECTION_ATTRIBUTES [connection]
  [node_name] /: "parameters_list"
```

where:

NATCICS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. See *Access File Attributes* on page 1698.

node_name

Is a name of the NODE definition in the ODIN configuration file.

userid

Is the primary authorization ID by which you are known to the Adapter for NATURAL CICS Transactions.

password

Is the password associated with the primary authorization ID.

parameters_list

The following parameters are supported (some required; some optional):

application_ID is the application ID of the CICS region. This parameter is optional.

mirror_name is the name of the mirror transaction. This value overwrites the MIRROR value provided when the Listener is configured. This parameter is optional.

communication is the communication protocol being used. The options are: TCP/IP, TCP 6.2 (AnyNET), and LU 6.2.

natnuc_name is the name of the NATURAL nucleus in CICS. This value is NATURAL CICS-dependent. This parameter is required.

natproxy_name is the name of proxy program in CICS. The default value is AASNATC. This value overwrites the PROXY value provided when the Listener is configured. This parameter is required.

Configuring the Adapter on z/OS

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for NATURAL CICS Transaction on z/OS

The CICS Transaction for Natural adapter is under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Communication Protocol

Select one of the following: *TCP/IP* or *EXCI*. Subsequent options vary slightly depending on this selection.

This selection is reflected in the Communication field on the second configuration pane.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Host name

For TCP/IP, enter the name of the host machine.

Application ID

For EXCI, enter the application ID of the CICS region.

Port

For TCP/IP, enter the port number.

Mirror

Is the name of the mirror transaction. The default value is EXCI.

This value overwrites the value provided in the Listener mode.

Proxy

Enter the name of a proxy program in NATURAL CICS. The default value is AASNATC. This value overwrites the PROXY parameter value provided when the Listener is configured.

Natural driver

Enter the name of the NATURAL nucleus in CICS. This value is NATURAL CICS-dependent.

Security

For EXCI, Trusted security is supported under the following conditions: the server must be running with security OPSYS and IRC (Inter Region Communications) must be active in the CICS region. The adapter then connects to the CICS region using the credentials of the operating system user impersonated by the server data access agent.

For TCP/IP, there are two methods by which a user can be authenticated when connecting to the Adapter for NATURAL CICS Transactions:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the adapter, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to CICS Transactions, at connection time, for authentication.

User

Is the user name by which you are known to the adapter. This field is only displayed when the security mode of the server is non-trusted.

Password

Is the password associated with the user name. This field is only displayed when the security mode of the server is non-trusted.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually on z/OS

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the server global profile, edasprof.prf, for all users:

```
ENGINE NATCICS SET CONNECTION_ATTRIBUTES [connection]
  [node_name]/userid,password: "parameters_list"
```

Password passthru. The user ID and password are explicitly specified for each connection and passed to CICS Transactions, at connection time, for authentication.

```
ENGINE NATCICS SET CONNECTION_ATTRIBUTES [connection]
  [node_name] /: "parameters_list"
```

Trusted. The adapter connects to an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE NATCICS SET CONNECTION_ATTRIBUTES [connection]
  [node_name] /: "parameters_list"
```

where:

`NATCICS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION=connection_name value. See *Access File Attributes* on page 1698.

`node_name`

Is a name of the NODE definition in the ODIN configuration file.

userid

Is the primary authorization ID by which you are known to the Adapter for NATURAL CICS Transactions.

password

Is the password associated with the primary authorization ID.

parameters_list

The following parameters are supported (some required; some optional):

application_ID is the application ID of the CICS region. This parameter is optional.

mirror_name is the name of the mirror transaction. This value overwrites the MIRROR value provided when the Listener is configured. This parameter is optional.

communication is the communication protocol being used.

natnuc_name is the name of the NATURAL nucleus in CICS. This value is NATURAL CICS-dependent. This parameter is required.

natproxy_name is the name of proxy program in CICS. The default value is AASNATC. This value overwrites the PROXY value provided when the Listener is configured. This parameter is required.

Managing NATURAL CICS Transactions Metadata

When the server invokes a transaction or procedure, it needs to know how to build the request, what parameters to pass, and how to format an answer set from the response. For each transaction the server will execute, you must create a synonym that describes the layout of the request/response area.

Creating Synonyms

Synonyms define unique names (or aliases) for each transaction or procedure that is accessible from the server. Synonyms are useful because they hide the underlying transaction or procedure from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows the input parameters and the response layout to be moved while allowing client applications to continue functioning without modification. For example, moving a transaction or procedure from a test region to production. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

CREATE SYNONYM is based on Local, Parameters, and Global data areas retrieved from the NATURAL ADABAS system file. These areas describe input and output buffers processed by NATURAL programs. If the NATURAL program does not have parameters, or if there are no NATURAL data areas that describe buffers, the CREATE SYNONYM command creates only template Master and Access Files.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for NATURAL CICS Transactions

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Natural System/User file with program libraries:

Data base ID

The database ID.

File number

A number that identifies the Natural System system file.

File password

If set, the password associated with the Natural system file.

Mask for library names

Natural library with Natural programs.

Enter a string for filtering library names, inserting the wildcard character (%) at the beginning and/or end of the string.

For example, enter ABC% to display libraries whose names begin with the letters ABC; %ABC to display libraries whose names end with the letters ABC; %ABC% to display libraries whose names contain the letters ABC at the beginning, middle, or end; or % to display all libraries.

Mask for program names

The mask for returning a list of program name from which to choose. Enter a string for filtering program names, inserting the wildcard character (%) at the beginning and/or end of the string.

Program names

Choose a program from the drop-down list.

No data area available

Check this box if your data is defined in either a local data area or a parameter data area. For more information, see the Software AG documentation on data area.

Note: If you check this box, you are re-prompted for database, file and password information on the next Create Synonym pane.

Leave this box unchecked if you are not using defined data.

Natural System/User file with data area definitions:

Data base ID

The database ID.

File number

A number that identifies the Natural System user file.

File password

If set, the password associated with the Natural system file.

Mask for the library names

The mask for returning a list of libraries from which to choose the one containing local/global data areas that describe the program's parameters.

Mask for the Data Area names

Enter the mask for returning a list of the data areas from which you will chose the data area defining the program's parameters.

Natural library

Select a library from the drop-down list.

Synonym name

The name of the synonym. You can retain the current name or change it.

Changing a synonym name enables you to manage multiple synonym versions to reflect, for example, multiple environments, or synonyms with different application logic such as different sets of Master File DEFINE attributes.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Input Data Area/Output Data Area

Specifies which data areas are for input parameters and which are for output parameters. You can specify the same data area as the source of both the input and output parameters.

Example: Generating a Synonym

The following example includes a NATURAL program and local data areas, with corresponding metadata.

IWAYIVPD NATURAL program:

```
DEFINE DATA
LOCAL USING NATTSTI /* INPUT PARMS FOR THE PROGRAM
LOCAL USING NATTSTO /* OUTPUT RECORD STRUCTURE
LOCAL
 1 #FUNC-TYPE
  2 #FUNC-GT (A2) INIT<'GT'>
  2 #FUNC-PT (A2) INIT<'PT'>
  2 #FUNC-LC (A2) INIT<'LC'>
  2 #FUNC-LI (A2) INIT<'LI'>
 1 #REQUEST-PARMS
  2 #FUNCTION (A2) /*GT,PT,LC,LI
  2 #OFFSET (I2) /*DATA OFFSET OF INPUT/OUTPUT
  2 #LENGTH (I2) /*LENGTH OF DATA TO GET OR PUT
  2 #RESPONSE-CODE (I4)
 1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 FIRST-NAME
  2 NAME
  2 MAR-STAT
  2 SEX
  2 BIRTH
  2 DEPT
  2 JOB-TITLE
  2 CURR-CODE(1:5)
  2 SALARY(N9/1:5)
 1 #ERROR-PARMS
  2 #NATPROG (A8)
  2 #NATMSG (A65)
  2 #NATERR (A7)
END-DEFINE

/* USE LI FUNCTION TO THE GET LENGTH OF INPUT PARAMETERS */
MOVE #FUNC-LI TO #FUNCTION
CALL 'AASSUBC' #REQUEST-PARMS
IF #LENGTH LT 16 /*REQUIRED FOR THIS PROGRAM*/
THEN TERMINATE
END-IF
```

```
/* USE GET FUNCTION TO RETRIEVE INPUT DATA PARMS */
MOVE 8 TO #LENGTH
MOVE 0 TO #OFFSET
MOVE #FUNC-GT TO #FUNCTION
CALL 'AASSUBC' #FUNCTION PERSONNEL-ID-FROM
CALL 'AASSUBC' #FUNCTION PERSONNEL-ID-TO
MOVE 0 TO #OFFSET
MOVE 145 TO #LENGTH
MOVE #FUNC-PT TO #FUNCTION
FIND ALL EMPLOY-VIEW WITH
  PERSONNEL-ID = PERSONNEL-ID-FROM THRU PERSONNEL-ID-TO
 IF NO RECORDS FOUND
  MOVE *PROGRAM TO #NATPROG
  MOVE ' REQUESTED EMPLOYEE NUMBERS NOT IN THE DATABASE' TO #NATMSG
  MOVE 80 TO #LENGTH
  MOVE #FUNC-PT TO #FUNCTION
  CALL 'AASSUBC' #FUNCTION #NATPROG
  TERMINATE
 END-NOREC
 /* CREATE OUTPUT RECORD
 MOVE BY NAME EMPLOY-VIEW TO OUTPUT-RECORD
 /* SEND OUPPUT RECORD
 CALL 'AASSUBC' #FUNCTION OUTPUT-RECORD.PERSONNEL-ID
END-FIND

ON ERROR
 MOVE *PROGRAM TO #NATPROG
 DECIDE FOR FIRST CONDITION
  WHEN *ERROR-NR = 1106
   MOVE ' EMPLOYEE NUMBER IS TOO LARGE. 8 BYTES IS '
   TO #NATMSG
   MOVE 'THE MAX' TO #NATERR
  WHEN *ERROR-NR = 3061
   MOVE ' INVALID EMPLOYEE NUMBER RANGE SPECIFIED '
   TO #NATMSG
   MOVE ' ' TO #NATERR
  WHEN NONE
   MOVE ' HAS DETECTED THE FOLLOWING ERROR NUMBER: '
   TO #NATMSG
   MOVE *ERROR-NR TO #NATERR
 END-DECIDE
 MOVE 80 TO #LENGTH
 CALL 'AASSUBC' #FUNCTION #NATPROG
 TERMINATE
END-ERROR
END
```

NATURAL local data areas:

```
Local IWAYTSTI Library SYSTEM DBID 3 FNR 9
I T L Name                  F Leng Index/Init/EM/Name/Comment
    1 INPUT-PARMS
    2 PERSONNEL-ID-FROM     A 8
    2 PERSONNEL-ID-TO       A 8

Local IWAYTSTO Library SYSTEM DBID 3 FNR 9
I T L Name                  F Leng Index/Init/EM/Name/Comment
    1 OUTPUT-RECORD
    2 PERSONNEL-ID          A 8
    2 FIRST-NAME            A 20
    2 NAME                  A 20
    2 MAR-STAT              A 1
    2 SEX                   A 1
    2 BIRTH                 D
    2 DEPT                  A 6
    2 JOB-TITLE             A 25
    2 CUR-CODE              A 3    (1:5)
    2 SALARY                N 9    (1:5)
```

Generated Master File: IWAYIVPD

```
FILENAME=IWAYIVPD, SUFFIX=NATCICS , CODEPAGE=37, $
 SEGMENT=SEG1, SEGTYPE=S0, $
  GROUP=INPUT_PARMS, ALIAS=E1, USAGE=A16, ACTUAL=A16, $
   FIELDNAME=PERSONNEL_ID_FROM, ALIAS=E2, USAGE=A8, ACTUAL=A8, $
   FIELDNAME=PERSONNEL_ID_TO, ALIAS=E3, USAGE=A8, ACTUAL=A8, $
 SEGMENT=SEG11, SEGTYPE=S0, PARENT=SEG1, $
  GROUP=OUTPUT_RECORD, ALIAS=E1, USAGE=A149, ACTUAL=A145, $
   FIELDNAME=PERSONNEL_ID, ALIAS=E2, USAGE=A8, ACTUAL=A8, $
   FIELDNAME=FIRST_NAME, ALIAS=E3, USAGE=A20, ACTUAL=A20, $
   FIELDNAME=NAME, ALIAS=E4, USAGE=A20, ACTUAL=A20, $
   FIELDNAME=MAR_STAT, ALIAS=E5, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=SEX, ALIAS=E6, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=BIRTH, ALIAS=E7, USAGE=P7, ACTUAL=P4, $
   FIELDNAME=DEPT, ALIAS=E8, USAGE=A6, ACTUAL=A6, $
   FIELDNAME=JOB_TITLE, ALIAS=E9, USAGE=A25, ACTUAL=A25, $
   FIELDNAME=CUR_CODE_POSN, ALIAS=E10, USAGE=A15, ACTUAL=A15, $
   FIELDNAME=SALARY_POSN, ALIAS=E11, USAGE=A45, ACTUAL=A45, $
   DEFINE BIRTH_DAT/YYMD=BIRTH - 694324; $
SEGMENT=SEG2,SEGTYPE=S0,PARENT=SEG11,OCCURS=5,POSITION=CUR_CODE_POSN, $
  FIELDNAME=CUR_CODE, ALIAS=E12, USAGE=A3, ACTUAL=A3, $
SEGMENT=SEG3,SEGTYPE=S0,PARENT=SEG11,OCCURS=5,POSITION=SALARY_POSN, $
  FIELDNAME=SALARY, ALIAS=E13, USAGE=P10, ACTUAL=Z9, $
```

Generated Access File: IWAYIVPD

```
SEGNAME=SEG1, CONNECTION=CON3, TRANSACTION=SYSTEM:IWAYIVPD, FLOAT=IBM,
 INTEGER=BigEndian, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Access File Attributes

Attribute	Description
SEGNAME	The name of the Master File segment that describes the stored procedure's input parameters. If the stored procedure does not have input parameters, the synonym generation process creates a dummy segment.
CONNECTION	The connection_name as previously specified in a SET CONNECTION_ATTRIBUTES command. The default connection is used if this parameter omitted.
TRANACTION	The names of the NATURAL library and the program to be executed.

NATURAL Data Buffer Processing API

This topic describes the flow control and implemented functions and response codes for the NATURAL data buffer processing API.

Flow of control:

NATCICS Adapter -> Proxy transaction (AASNATC) -> CICS NATURAL nucleus -> NATURAL program <-> AASSUBC. (Note that in the illustration in *Generating a Synonym* on page 1694. IWAYIVPD is an example of an AASNATC proxy transaction.)

```
Communication with the Data Mover program requires the following control
block:
 1 #REQUEST-PARMS
 2 #FUNCTION      (A2) One of the following: GT,PT,LC,LI
 2 #OFFSET        (I2)
 2 #LENGTH        (I2) Length of data for GT or PT functions
 2 #RESPONSE-CODE (I4)
```

The implemented functions are:

- **GT.** Get input data by the offset. Only on the first call #OFFSET must be set to 0. Data Mover program uses #OFFSET to keep the cursor position in the data buffer. The length of the requested data must be provided for each call in the #LENGTH parameter. It should match the length of the area provided to receive the input data. For example:

```
1 #EMP-NUM1   (A8)
1 #EMP-SALARY (A6)
.................
MOVE 'GT' TO #FUNCTION
MOVE 8 TO #LENGTH
MOVE 0 TO OFFSET
CALL 'AASSUBC' #FUNCTION #EMP-NUM1
MOVE 6 TO #LENGTH
CALL 'AASSUBC' #FUNCTION #EMP-SALARY
```

- **PT.** Put output data by the offset. Only on the first call #OFFSET must be set to 0. Data Mover program uses #OFFSET to keep the cursor position in the data buffer. The length of the stored data must be provided for each call in the #LENGTH parameter. It should match the length of the area containing output data. For example:

```
1 #EMP-NUM1 (A8) INIT <'12345678'>
1 #EMP-SALARY (A6) INIT <' 65000'>
.................
MOVE 'PT' TO #FUNCTION
MOVE 8 TO #LENGTH
MOVE 0 TO OFFSET - FIRST CALL ONLY
CALL 'AASSUBC' #FUNCTION #EMP-NUM1
MOVE 6 TO #LENGTH
CALL 'AASSUBC' #FUNCTION #EMP-SALARY
```

- **LI.** Get length of the input data. Updates the #LENGTH field with the total length of all input data. No other parameters are needed. For example:

```
MOVE 'LI' TO #FUNCTION
CALL 'AASSUBC' #FUNCTION
```

- **LC.** Get length of the COMMAREA used to send/receive data. Updates the #LENGTH field with the COMMAREA length. No other parameters are needed. For example:

```
MOVE 'LC' TO #FUNCTION
CALL 'AASSUBC' #FUNCTION
```

Implemented response-codes:

0 - operation completed successfully.

4 - end-of-data. Cursor positioned beyond the input buffer. Indicates end of input data.

8 - end-of-buff. Cursor position plus requested length for PT operation exceeds the COMMAREA length. Indicates COMMAREA overflow.

Note:

- Always use the first field of a group when calling Data Mover programs. Notice that all calls in the previous examples are made with #FUNCTION not #REQUEST-PARMS.
- All input parameters must be processed before making a PT call. Otherwise, data will be overwritten.

Invoking a NATURAL CICS Transaction

To invoke a NATURAL CICS transaction, you issue a Data Manipulation Language (DML) request, also known as a TABLE command, an SQL SELECT statement, or an EX command. (DML is the WebFOCUS internal retrieval language.) For information about the Data Manipulation Language, see the *Stored Procedure Reference* manual.

Syntax: How to Invoke a NATURAL CICS Transaction Using EX

```
EX [app_name_space/]synonym 1=parm1_val,..., N=parmN_val

EX [app_name_space/]synonym parm1_name=parm1_val,... ,
                            parmN_name=parmN_val

EX [app_name_space/]synonym [, parm1_val [,...[, parmN_val]]]
```

where:

app_name_space

Is the apps directory name under which the synonyms are stored. This value is optional.

synonym

Is the user friendly name of a repository entry that represents the object to be executed in the vendor environment.

parm_name

Is the name of the parameter taken from the input metadata description.

1=parm1_val

N=parmN_val

parm1_name

parm1_val

parmN_val

Are the parameter values that match the input metadata description.

Note:

- Consecutive commas denote missing parameters in the positional form of the request.
- Values containing special characters (<equal sign> <space> <comma>) must be encapsulated in double or single quotation marks.

Example: Invoking the Natural Program IWAYIVPD Using EX

```
SET EXORDER=PGM/FEX
EX IWAYIVPD 11111111, 11211111
```

Syntax: How to Invoke a NATURAL CICS Transaction Using TABLE

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the NATURAL CICS transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

IF/WHERE

Is used if you want to pass values to input parameters.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Example: Invoking the Natural Program IWAYIVPD Using TABLE

```
TABLE FILE IWAYIVPD
 PRINT PERSONNEL_ID NAME BIRTH SALARY
 IF PERSONNEL_ID_FROM EQ '11111111'
 IF PERSONNEL_ID_TO EQ '11211111'
 IF SALARY NE 0
END
```

Syntax: How to Invoke a NATURAL CICS Transaction Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the NATURAL CICS transaction you want to invoke.

parameter

Is the name of a parameter whose values you want to display. You can specify input parameters, output parameters, or input and output parameters.

If the transaction does not require parameters, enter an * in the syntax. This displays a dummy segment, created during metadata generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters.

WHERE

Is used if you want to pass values to input parameters.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an input parameter to which you want to pass a value.

value

Is the value you are passing to an input parameter.

Example: **Invoking the Natural Program IWAYIVPD Using SQL SELECT**

```
SQL
 SELECT PERSONNEL_ID, NAME, BIRTH, SALARY
  FROM IWAYIVPD
  WHERE PERSONNEL_ID_FROM = '11111111' AND
  PERSONNEL_ID_TO = '11211111' AND
  SALARY != 0;
END
```

Chapter 69

Using the Adapter for Netezza

The Adapter for Netezza allows applications to access specific Netezza data sources. The adapter converts application requests into Netezza calls and returns optimized answer sets to the requesting application.

One implementation is available, JDBC.

In this chapter:

Preparing the Netezza Environment

In order to use the Adapter for Netezza, you must install the Netezza driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the Netezza driver files by adding them to the environment variable CLASSPATH before server startup.

   Note: If Db2 v11 and Netezza are installed on the same machine, the Netezza jar file must be listed first in the CLASSPATH.

   For example, to set a UNIX or IBM i location for Netezza JDBC driver files:

   ```
   CLASSPATH=/rdbms/netezza_jar/netezza-jdbc.jar:
   /rdbms/netezza_jar/bcprov-jdk14-122.jar:$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\<netezza_jdbc_driver_files>;%CLASSPATH
   %
   ```

See the Netezza documentation for specific file names and their locations.

Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server's EDAPRINT. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Unicode Support in Netezza

Netezza Unicode data in NCHAR/NVARCHAR fields is supported when the reporting server is configured for Unicode with code page 65001.

Configuring the Adapter for Netezza

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Netezza data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Netezza data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Netezza JDBC

The *Netezza* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Netezza data source.

Driver Name

Name for the JDBC driver.

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLNEZ SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLNEZ`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is a logical name used to identify this particular set of attributes.

Note that one blank space is required between connection and URL.

`URL`

Is the URL to the location of the Netezza data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the Netezza Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLNEZ SET CONNECTION_ATTRIBUTES CON1
  'jdbc:xxxxxxx://hostname:port/datasource
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLNEZ SET DEFAULT_CONNECTION connection
```

where:

`SQLNEZ`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects CON1 as the default connection.

```
ENGINE SQLNEZ SET DEFAULT_CONNECTION CON1
```

Managing Netezza Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLNEZ to identify the Adapter for Netezza.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLNEZ [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLNEZ

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using Netezza, you must locally install the RDBMS Netezza Driver.

The Server can access third-party database tables across the JDBC network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Netezza

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Select Database

Select a database from the Select database drop-down list, which lists all databases in the current DBMS instance.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Netezza synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Note: The sample data, table nf29004, does not exist.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLNEZ ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Netezza table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Netezza Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Netezza Environment

The Adapter for Netezza provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Netezza.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLNEZ SET TIMEOUT {nn|0}
```

where:

`SQLNEZ`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

0

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLNEZ SET PASSRECS {ON|OFF}
```

where:

SQLNEZ

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Controlling HOLD DBMS Creation

An extension of the HOLD AS *dbms_name* FORMAT SQLNEZ syntax enables you to exercise more precise control over the creation of HOLD DBMS tables.

Syntax: How to Control DBMS Creation

```
HOLD [AS dbms_name] FORMAT SQLNEZ [CONNECTION conn_name]
```

where:

dbms_name

Is the DBMS table to create. It may be a one, two, or three part name, using the separator appropriate to the DBMS, typically a dot (.).

conn_name

Is the DBMS connection name. When multiple DBMS connections have been configured and are in use, *conn_name* specifies which connection to use.

Note: If a table exists with the same name and connection, it will be dropped.

Example: Creating a Netezza Table

The following command creates a Netezza table named DBCS_NEZ using connection CON1:

```
ON TABLE HOLD AS DBCS_NEZ FORMAT SQLNEZ CONNECTION CON1
```

Netezza Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Optimizing Non-equality WHERE-based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

`LEFT_OUTER`

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

`hostfile`

Is the host Master File.

`AT`

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

`hfld1`

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

`tag1`

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

`WITH hfld2`

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

`MULTIPLE`

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

`UNIQUE`

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations. It always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: Conditions for WHERE-Based Outer Join Optimization

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left-Outer Join

The following request creates a left outer conditional join between two Netezza data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE baseapp/EQUIP AT CARS
TO ALL FILE baseapp/CARREC AT CARC
WHERE CARS NE CARC;
END
TABLE FILE baseapp/EQUIP
PRINT CARS CARC STANDARD
BY MODEL
END
```

The WebFOCUS request is translated to a single Netezza SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."CARS"(CHAR( 16)),T1."STANDARD"(CHAR( 40)),
T2."CARC"(CHAR( 16)),T2."MODEL"(CHAR( 24)) FROM ( EQUIP T1 LEFT
OUTER JOIN CARREC T2 ON (T1."CARS" <> T2."CARC") ) ORDER BY
T2."MODEL";
```

Chapter 70

Using the Adapter for Nucleus

The Adapter for Nucleus allows applications to access Nucleus data sources. The adapter converts application requests into native Nucleus statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Nucleus Environment

In order to use the Adapter for Nucleus, the Nucleus database and ODBC must be installed and configured and the path to the Nucleus ODBC libraries directory must be added to SYSTEM LIBRARY PATH. See the SandTechnology Nucleus(r) documentation for more information about requirements for ODBC installation and configuration.

Procedure: How to Set Up the Environment on Windows

On Windows, the Nucleus environment is set up during the installation of Nucleus.

Procedure: How to Set Up the Environment on UNIX

You can access Nucleus using the NUCINI environment variable and one of the following:

- $HOME/.odbc.ini file
- ODBCINI environment variable.

Point the NUCINI variable to the directory where SandTechnology ODBC(r) is installed. For example:

```
NUCINI=/usr/nucleus/odbc
export NUCINI
```

The ODBCINI variable holds the absolute path to the *.ini file, which describes the Nucleus data sources. For example:

```
ODBCINI=/usr/ibiuser/odbc/nucodbc.ini
export ODBCINI
```

Configuring the Adapter for Nucleus

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Nucleus database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Nucleus database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Nucleus Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Nucleus

The Nucleus adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Authorization ID to connect to the source.

Password

Password associated with the authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLNUC SET CONNECTION_ATTRIBUTES connection
DSN_name/userid,password
```

where:

`SQLNUC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the Nucleus Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`userid`

Is the primary authorization ID by which you are known to Nucleus.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the Nucleus DSN named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLNUC SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console's Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

  ```
  ENGINE SQLNUC SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

```
ENGINE SQLNUC SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLNUC SET DEFAULT_CONNECTION connection
```

where:

`SQLNUC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLNUC SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLNUC SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLNUC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Nucleus Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Nucleus data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Nucleus table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Nucleus

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Nucleus Data Type Support* on page 1737.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Nucleus synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLNUC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Nucleus table. The value assigned to this attributes can include the name of the owner (also known as schema) as follows: TABLENAME=[*owner*.] *table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection*

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Nucleus Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLNUC SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Nucleus Environment

The Adapter for Nucleus provides several parameters for customizing the environment and optimizing performance.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLNUC SET PASSRECS {ON|OFF}
```

where:

`SQLNUC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Nucleus Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 71

Using the Adapter for OData

The Adapter for OData is a REST-based adapter that integrates with RESTful Web Services built according to the OData (Open Data Protocol) specification (*https://www.odata.org/*).

The adapter supports v4 of the OData specification.

In this chapter:

Configuring the Adapter for OData

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The Adapter for OData is in the XML Based group.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for OData

Enter or select values for the following connection parameters to add a connection to the Adapter for Slack.

Connection Name

Is the logical name used to identify this particular set of connection attributes. The default is CON01.

Base URL

Is the URL to the OData service, for example, *https://services.odata.org/V4/Northwind/Northwind.svc/%20%20*.

Security

There are four methods by which a user can be authenticated when connecting to an OData Service provider:

- **Trusted.** Uses a KERBEROS protocol that may or may not require a Service Principal Name.
- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the OData Service, at connection time, for authentication.
- **OAuth.** Open Authorization grants limited access to user accounts on an HTTP service.
- **None.** No security. User ID and password are not required.

KERBEROS SPN

Is the Service Principal Name used by the client to uniquely identify an instance of a service, for example, mydaemon/foo:4761. This field appears when Trusted security is selected.

Token URL

Is the URL used for obtaining an Access Token to OData. This field appears when OAuth security is selected.

Additional Authentication Parameters

This field appears when OAuth security with the Authorization Code Grant Type is selected. Enter any additional parameters required for authentication.

Additional Token Parameters

This field appears when OAuth security with the Authorization Code Grant Type is selected. Enter any additional parameters required for token retrieval.

User

Is the primary authorization ID by which you are known to the OData Service. This field appears when Explicit security or OAuth security with the Password Grant Type is selected.

Password

Is the password associated with the primary authorization ID. This field appears when Explicit security or OAuth security with the Password Grant Type is selected.

OAuth Grant Type

Can be one of the following OAuth authentication types:

- Authorization code.
- Password.
- Client Credentials.

This field appears when OAuth security is selected.

Service Provider

OAuth configuration for certain Service Providers. This field appears when OAuth security is selected with the Authorization Code grant type.

Service URL

The URL to the Web Service when not configured as a REST or OData adapter connection.

Client ID

Is the client ID defined in the OData application. This field appears when OAuth security is selected.

Client Secret

Is the Client Secret defined in the OData application. This field appears when OAuth security is selected.

Authorization URL

URL used for OAuth Authorization to a specific application.

For example, the Authorization URL for the Google set of APIs is https://accounts.google.com/o/oauth2/auth.

Access Token

Is the value that identifies the user on whose behalf your OData application is acting. This field appears when OAuth security with the Authorization Code Grant Type is selected. Click *Get Access Token* to obtain this token.

In order for *Get Access Token* to complete successfully, the host name used to bring up the Reporting Server Web Console must match the host name set up for the Redirect URI in the OData application.

Refresh Token

Is the Refresh Token returned from the OAuth Token request. This field appears when OAuth security with the Authorization Code Grant Type is selected. The token is used for obtaining a new Access Token at the time a report is run accessing the OData Service for a specific application.

Select profile

Is the profile (server, group, user) in which to store these connection attributes. The default is the server profile, edasprof.prf.

Advanced Connection Options

Add Custom Headers

Check to open a text box in which you can enter Custom Header information required by the OData Service.

Select within Expand

When checked, select within expand OData syntax is not supported.

Advanced HTTP Connection Options

PROXY Server IP Address

Is the IP address of the proxy server.

PROXY Port

Is the port number on which the proxy server listens. The default port number is 80.

PROXY HTTPS Relative Path

When checked, the REST request will send the relative path rather than the absolute path when a proxy server is configured.

SSL Certificate

Is the location of a locally-stored user-provided server x.509 certificates file for SSL authentication. For trusted authentication, the trusted certificate file (trustedcertfile) points to a file of CA certificates in PEM format, as illustrated in the following syntax:

```
-----BEGIN CERTIFICATE-----
... (CA certificate in base64 encoding) ...
-----END CERTIFICATE-----
```

SSL Mutual Authentication

When checked, Mutual Authentication is enabled.

SSL Certificate Type

Select one of the following certificate types:

- **Trusted.** A trusted certificate file can contain several CA certificates. You can add text before, between, and after any certificate which is typically done to provide descriptions of each certificate.
- **Non-trusted.** Adds the parameters Key file, Pass phrase, and Label to the configuration pane. Provide an initial path browse value to the SSL Certificate field before browsing. For example:

  ```
  C:\, /abc/abc/...
  ```

SSL Certificate Key File

Is the private key used for creating the client X.509 certificate in PEM format. This option is used together with a certificate for a non-trusted connection. Provide an initial value to the SSL certificate key file field before browsing. For example:

```
C:\, /abc/abc/...
```

SSL Certificate Pass Phrase

Is the password used to unlock the key file. The value is needed only if the key file is encrypted.

SSL Certificate Label

Identifies a certificate in the file, if the file contains more than one certificate. If the label contains spaces, the label must be enclosed in double-quotes. For example:

```
"xxx yyy"
```

Managing OData Metadata

Create Synonym for the Adapter for OData creates the metadata used for WebFOCUS reporting against data returned from the REST API calls.

Procedure: How to Create Metadata for the Adapter for OData

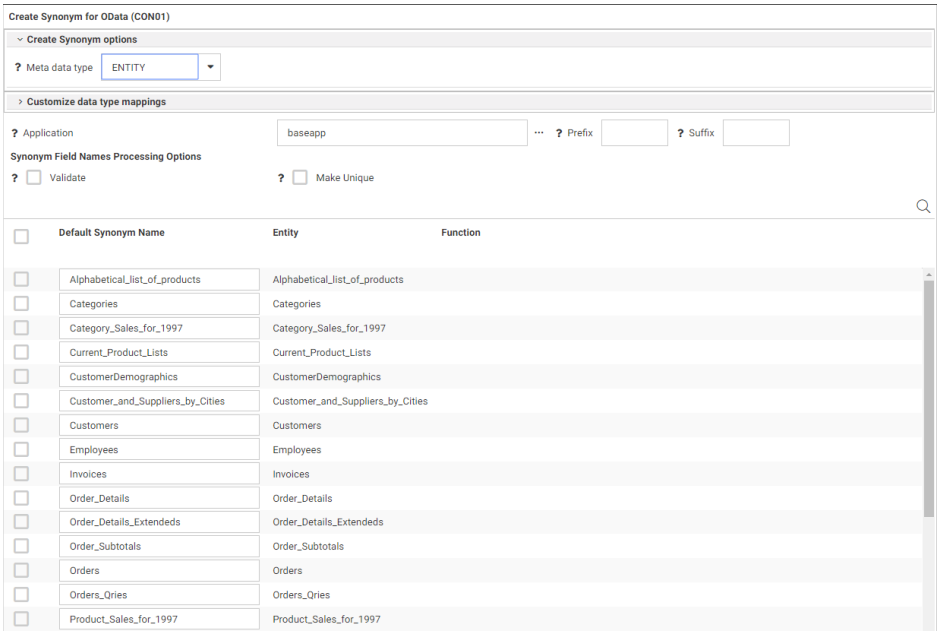
To create a synonym, you must have previously configured the adapter.

1. From the Web Console sidebar, click *Connect to Data*.

   The Connect to Data page opens.

2. On the Configured list, right-click a connection for the configured adapter and click *Show DBMS Objects*.

The Create Synonym for OData pane opens, as shown in the following image.



3. Select a Meta data type, either ENTITY or FUNCTION.
4. Enter a specific application in the Application field, or click the ellipsis button to the right of the field to select an application in which to store the metadata.
   a. Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.)

      This parameter ensures that names adhere to specifications. For more information, see *Validation for Special Characters and Reserved Words* on page 2743. When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

   b. Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

   c. Click *Customize data type mappings* to select whether to decompose date formats, assign geographic roles automatically, set the date order, and select the data type mappings for numeric decimal and alphanumeric columns.
5. Select the check boxes next to the entities or functions for which you want to create metadata.

You can change the name of the synonym to be created by typing over the default synonym name.

6. When you have made your selections, click the highlighted *Create Synonym* button on the ribbon.

The metadata is created and added under the specified application directory.

Example: Sample Synonym

The following is the generated Master File for the CustomerDemographics entity. The suffix value is ODATAV4:

```
FILENAME=M6ILO, SUFFIX=ODATAV4 , $
  SEGMENT=CUSTOMERDEMOGRAPHICS, SEGTYPE=S0, $
    FIELDNAME=CUSTOMERDEMOGRAPHICS, ALIAS=value, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=CUSTOMERTYPEID, ALIAS=CustomerTypeID, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERDEMOGRAPHICS, $
    FIELDNAME=CUSTOMERDESC, ALIAS=CustomerDesc, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERDEMOGRAPHICS, $
  SEGMENT=CUSTOMERS, SEGTYPE=S0, PARENT=CUSTOMERDEMOGRAPHICS, $
    FIELDNAME=CUSTOMERS, ALIAS=Customers, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=CUSTOMERDEMOGRAPHICS, $
    FIELDNAME=CUSTOMERID, ALIAS=CustomerID, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=COMPANYNAME, ALIAS=CompanyName, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=CONTACTNAME, ALIAS=ContactName, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=CONTACTTITLE, ALIAS=ContactTitle, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=ADDRESS, ALIAS=Address, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=CITY, ALIAS=City, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=REGION, ALIAS=Region, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=POSTALCODE, ALIAS=PostalCode, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=COUNTRY, ALIAS=Country, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=PHONE, ALIAS=Phone, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
    FIELDNAME=FAX, ALIAS=Fax, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=CUSTOMERS, $
```

The following is the generated Access File for the CustomerDemographics entity:

```
SEGNAME=CUSTOMERDEMOGRAPHICS,
  CONNECTION=CON01,
  TABLENAME=CustomerDemographics,
  OBJECT_URL=CUSTOMERTYPEID,
  EXTCALL=YES, $
 FIELD=CUSTOMERTYPEID,
    TYPE=ID,
    IN_URL=TRUE, $
SEGNAME=CUSTOMERS,
  OBJECT_URL=CUSTOMERID,
  EXTCALL=NO, $
 FOREIGN_KEY=Customers,
   PRIMARY_KEY_TABLE=Customer,
   FOREIGN_KEY_COLUMN=CustomerID,
   PRIMARY_KEY_COLUMN=CustomerID, $
 FIELD=CUSTOMERID,
    TYPE=ID,
    IN_URL=TRUE, $
```

Chapter 72

Using the Adapter for ODBC

The Adapter for ODBC allows applications to access ODBC data sources. The adapter converts application requests into native ODBC statements and returns optimized answer sets to the requesting application.

For details about ODBC-supported data sources, see the *Server Release Notes*.

In this chapter:

Preparing the ODBC Environment

In order to use the Adapter for ODBC, you must install the Microsoft 32-bit ODBC Driver Manager on the Windows system. The Adapter for ODBC accesses all ODBC drivers installed and defined in the 32-bit ODBC Driver Manager.

Note:

- If Information Builders offers an adapter for a specific data source type, you should use that adapter to access that type. The Adapter for ODBC does not support data source types for which a specific adapter already exists. For example, Information Builders provides the Adapter for Db2 to access Db2 databases, so you should not use the Adapter for ODBC to access Db2.

- If Information Builders does not currently offer a specific adapter for a data source type, you may use the Adapter for ODBC with a vendor-provided ODBC driver. Note that the adapter conforms to generic ODBC standards. Vendors, on the other hand, may have interpreted or extended those standards for their specific data source requirements. If this introduces an incompatibility, you may experience inconsistent behavior when using the adapter for that vendor's data source type. If this occurs please contact Customer Support Services for assistance. Information Builders will work with your enterprise to try and support your data access needs.

Configuring the Adapter for ODBC

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to the ODBC database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one ODBC database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the ODBC Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for ODBC

The ODBC adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

If a File DSN is used this must be the name of the file (for example, samplefdsn.dsn.)

Security

There are two methods by which a use can be authenticated when connecting to an ODBC data source:

- **Explicit.** The user ID and password are explicitly specified for each connection pass to the ODBC data source, at connection time, for authentication.
- **Trusted.** The adapter connects to the ODBC data source as a Windows login using the credentials of the operating system user impersonated by the server data access agent.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

File DSN parameters (optional)

The path to the ODBC File DSN that is defined on your Windows machine or network.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to ODBC, at connection time, for authentication.

For User/System DSN:

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES connection DSN_name/
userid,password
```

For File DSN:

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES connection
 FileDSN_name/userid,password:'FileDSN_path;'
```

Trusted authentication. The adapter connects to ODBC as a Windows login using the credentials of the Windows user impersonated by the server data access agent.

For User/System DSN:

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES connection DSN_name/,
```

For File DSN:

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES connection
 FileDSN_name/,:'filedsn=FileDSN_path;',
```

where:

`SQLODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the ODBC Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`FileDSN_`*`name`*

Is the name of ODBC File Data Source Name (DSN) you wish to access.

`userid`

Is the primary authorization ID by which you are known to ODBC.

`password`

Is the password associated with the primary authorization ID.

`FileDSN_`*`path`*

Is the path to the ODBC File DSN, which may reside on the hard drive or on the network.

Example: Declaring Connection Attributes

For User/System DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the ODBC DSN named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

For File DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 via the ODBC File DSN named SAMPLESERVER.dsn with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES CON2
SAMPLESERVER.dsn /R729999B,1525B6F3331C2FB3:'filedsn=C:\ SAMPLESERVER.dsn;'
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
```

When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

```
CONNECTION=DSN_A
```

If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

```
ENGINE SQLODBC SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLODBC SET DEFAULT_CONNECTION connection
```

where:

`SQLODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLODBC SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLODBC SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing ODBC Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the ODBC data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLODBC to identify the Adapter for ODBC.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLODBC [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLODBC

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using ODBC, you must locally install the RDBMS ODBC Driver.

The Server can access third-party database tables across the network ODBC. You must provide a system data source name and possibly a user ID and/or password for the database tables you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each ODBC table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.

2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for ODBC

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 1766.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for ODBC synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=ODB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the ODBC table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local ODBC database server. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
DBSPACE	Optional keyword that indicates the storage area for the table. For example: *datasource.tablespace* DATABASE *datasource*
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

If you wish to use the Adapter for ODBC to access a data source, first check whether Information Builders already provides an adapter for that type. To find out which adapters are currently available, see *Supported Adapters*.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the ODBC Environment

The Adapter for ODBC provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to ODBC.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLODBC SET TIMEOUT {nn|0}
```

where:

`SQLODBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

0

Represents an infinite period to wait for a response.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLODBC SET PASSRECS {ON|OFF}
```

where:

SQLODBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE SQLODBC SET ISOLATION {RU|RC|RR|SE}
```

where:

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

ODBC Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 73

Using the Adapter for Oracle

The Adapter for Oracle allows applications to access Oracle data sources. The adapter converts data or application requests into native Oracle statements and returns optimized answer sets to the requesting program.

In this chapter:

Preparing the Oracle Environment

The Adapter for Oracle minimally requires the installation of the Oracle Client. The Oracle Client allows you to connect to a local or remote Oracle database server.

Make sure that the client shared library, libclntch, was generated on UNIX and z/OS USS.

If you are using Oracle 11g or 10g and wish to take advantage of the adapter's support for Unicode, see *Configuring Oracle for Unicode* on page 1772.

Procedure: How to Prepare the Oracle Environment on Windows

On Windows, the Oracle environment is set up during the installation of Oracle.

Procedure: How to Prepare the Oracle Environment on UNIX

You can issue the following export commands in the edastart file. (Alternatively, you can issue them in a separate shell file, a database profile, or a user profile.)

1. On the Web Console menu bar, select *Workspace*, then *Configuration*. In the Configuration pane, expand the *Configuration Files* folder and choose *Server Startup Script* to edit the edastart file. Place the export commands immediately following the shell setting. For example:

   ```
   #!/bin/ksh
   export ORACLE_SID=orac
   export ORACLE_HOME=/usr/oracle/orac
   . . .
   ```

2. Specify the Oracle database instance to access using the UNIX environment variable $ORACLE_SID. For example:

   ```
   ORACLE_SID=orac
   export ORACLE_SID
   ```

3. Specify the location of the Oracle database you wish to access using the UNIX environment variable $ORACLE_HOME. For example, to set the home directory for the Oracle software to /usr/orac, specify:

   ```
   ORACLE_HOME=/usr/oracle/orac
   export ORACLE_HOME
   ```

4. Specify the path to the Oracle shared library using the appropriate platform-dependant UNIX environment variable, such as $LD_LIBRARY_PATH for Sun Sparc, $SHLIB_PATH for HP-UX, and $LIBPATH for AIX.

   This must be done only for an unsecured server. For example:

   ```
   LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
   export LD_LIBRARY_PATH
   ```

   Note: If the server is running with security on, the LD_LIBRARY_PATH variable is ignored. In this case, you must use IBI_LIBPATH.

Procedure: How to Prepare the Oracle Environment on z/OS

The configuration for Oracle on z/OS requires that EDASTART JCL is used to allocate Oracle specific variables. This can be done using the EDAENV ddname in the JCL:

```
ORACLE_SID=ORAT
ORACLE_HOME=/usr/lpp/orac
LIBPATH=$ORACLE_HOME/lib
```

If you are running a secure server, then your Oracle run-time load and message libraries must be APF-authorized in order to avoid the following error message:

```
ORA-01019: unable to allocate memory in the user side
```

Another option is to implement an internal security mechanism by placing the following parameter in EDASERVE DDNAME.

```
APFAUTH=INTERNAL
```

With this setting, the server calls Oracle in a non-authorized state so that Oracle libraries do not require any special APF authorization (except when the Oracle documentation states differently).

Syntax: **How to Prepare the Oracle Environment on OpenVMS**

When a site creates an Oracle SID, the Oracle software automatically generates a DCL setup script so users and products such as the Reporting Server, can properly invoke the environment.

The specification for the location of the Oracle setup file is:

```
$@disk:[oracle_root.DB_oraclesiddb]ORAUSER_oraclesiddb.COM
```

where:

disk

Is the disk on which Oracle is installed.

oracle_root

Is the root directory of the Oracle installation.

oraclesiddb

Is the name of the Oracle database. This name does not need to match the Oracle SID.

EDAENV.COM is an optional OpenVMS file that is invoked at server start up to issue DCL commands such as calls to DBMS setup files. Due to the use of symbols and job logicals in Oracle, the only proper way to invoke the Oracle setup file is by using the EDAENV.COM file.

By default, EDAENV.COM does not exist and must be manually created in the [.BIN] directory of EDACONF. In this case, and in its simplest form, EDAENV.COM will contain a single line of syntax that specifies the call to the Oracle setup file.

Example: **Specifying the $ORACLE_SID on OpenVMS**

```
$@3$DKB0:[ORACLE.DB_ORAC]ORAUSER_ORAC.COM
```

Connecting to a Remote Oracle Database Server

Using the standard rules for deploying the Oracle Client, the server supports connections to:

- Local Oracle database servers.
- Remote Oracle database servers. To connect to a remote Oracle database server, the Oracle tnsnames.ora file on the source machine must contain an entry pointing to the target machine and the listening process must be running on the target machine.

Once you are connected to an Oracle database server, that server may define Oracle DATABASE LINKs that can be used to access Oracle tables on other Oracle database servers.

XA Support

Read/write applications accessing Oracle data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see *XA Support* on page 2689.

Configuring Oracle for Unicode

The adapter supports Unicode data in Oracle release 10g or higher databases that have been configured with the NLS_CHARACTERSET parameter set to UTF8. You must set the NLS_LANG environment variable in the edastart file or in a separate shell file.

Note:

- Oracle Unicode data in NCHAR/NVARCHAR fields is supported when the reporting server is configured for Unicode.
- The Oracle environment variable ORA_NCHAR_LITERAL_REPLACE controls transparent data conversion into NCHAR/NVARCHAR fields. This variable usually remains unset to maintain better performances.

  However, ORA_NCHAR_LITERAL_REPLACE must be set to TRUE in the following rare cases:

  - SQL Passthru is used for input into Oracle NCHAR/NVARCHAR fields (INSERT/UPDATE/WHERE clause) . The literal must also be in N'<string>' notation.
  - TABLE FILE contains IF/WHERE clause(s) with NCHAR/NVARCHAR field(s).

Syntax: **How to Set the Environment Variable for NLS_LANG**

To set the NLS_LANG environment variable to support Unicode, use the following syntax

```
NLS_LANG = language_territory.characterset
```

where:

language

Is the selected language.

territory

Is the name of the country associated with the selected language.

characterset

Is the value of the NLS_CHARACTERSET variable that is set in the Oracle database. For Unicode, this is always UTF8.

Example: **Setting the NLS_LANG Variable for American English UTF-8**

For American English UTF-8, the following setting is required:

```
NLS_LANG=American_America.UTF8
```

Configuring the Adapter for Oracle

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an Oracle database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Oracle database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Oracle Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Oracle

The *Oracle* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Is the logical name used to identify this particular set of connection attributes. There is no default connection name. You must provide a value.

TNS name

Service (TNS) name used as a connect descriptor to an Oracle database server across the network. It must point to a valid entry in the tnsnames file.

Connection to a local database server can be entered as <local>. Note that <local> may not have a match in the tnsnames.ora file.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.

  Note: If you have Oracle Proxy Authentication set in your Oracle instance, as described in the Oracle vendor documentation, you can use explicit authentication to configure a connection that uses Oracle Proxy Authentication. To configure this type of authentication, enter the following syntax in the User field:

  srv_userid[*proxy_userid*]

  where:

  srv_userid is the user ID for the server connection to Oracle.

  proxy_userid is the user ID for the proxy server.

- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication.

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES connection TNS_name/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Oracle, at connection time, for authentication.

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES connection TNS_name/
```

Trusted authentication. The adapter connects to Oracle as an operating system login using the credentials of the operating system user impersonated by the server data access agent.

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES connection TNS_name/,
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is a logical name used to identify this particular set of attributes.

Note that one blank space is required between *connection* and TNS_*name*.

`TNS_name`

Is the Service (TNS) name used as a connect descriptor to an Oracle database server across the network. It must point to a valid entry in the tnsnames file. Connection to a local database server can be entered as <local>. Note that <local> may not have a match in the tnsnames.ora file.

userid

Is the primary authorization ID by which you are known to Oracle.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allows the application to access the Oracle database server named SAMPLESERVER, defined in tnsnames.ora as SAMPLESERVER, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the local Oracle database server and does not use tnsnames:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES <Local> <Local>/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Oracle database server named SAMPLESERVER using Password Passthru Authentication:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER SAMPLESERVER/
```

The following SET CONNECTION_ATTRIBUTES command connects to a local Oracle database server using operating system authentication:

```
ENGINE SQLORA SET CONNECTION_ATTRIBUTES SAMPLESERVER SAMPLESERVER/,
```

Reference: Updating the Connection String

The syntax for the SET CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

  ```
  ENGINE SQLORA SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

  ```
  ENGINE SQLORA SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
  ```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLORA SET DEFAULT_CONNECTION connection
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLORA SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLORA SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Oracle Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Oracle data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Oracle table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 1792.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Oracle

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Oracle Data Type Support* on page 1788.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Oracle synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLORA ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=ORA901,KEYS=1, WRITE=YES,$
```

Reference: Mapping Oracle Comments into a Synonym

When you generate a synonym for an Oracle table, the adapter maps each:

- Oracle table comment from the Oracle data dictionary to a Remark attribute in the Master File of the synonym.
- Oracle column comment from the Oracle data dictionary to a Description attribute in the Master File of the synonym.

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.

Keyword	Description
TABLENAME	Identifies the Oracle table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*[*@databaselink*]
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local Oracle database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. This attribute requires the columns that constitute the key to be described first in the Master File. See the KEY attribute below, for information about specifying the key fields without having to describe them first in the Master File. **Note:** If the table does not have the primary key, then the first unique index may be taken instead.
KEY	Specifies the columns that participate in the primary key without having to describe them first in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Accessing Multiple Database Servers in One SQL Request

To access a remote Oracle table using DATABASE LINKs, the following conditions must exist:

- The Oracle database server to which you are connected must have a valid DATABASE LINK defined.
- The TABLENAME attribute in the Access File for the Oracle table to be queried must have the following format:

  TABLENAME=[*owner.*]*table@databaselink*

where:

owner

Is the user ID by default. It can consist of a maximum of 30 characters. Oracle prefers that the value be uppercase.

table

Is the name of the table or view. It can consist of a maximum of 30 characters.

databaselink

Is the valid DATABASE LINK name defined in the currently connected Oracle database server.

This format for TABLENAME can be placed in the Access File manually or using the CREATE SYNONYM command. For example:

```
CREATE SYNONYM filename FOR owner.table@databaselink DBMS SQLORA
```

Once you have met the above conditions, all requests for the table will be processed on the remote Oracle database server specified using the DATABASE LINK name. Using this method is another way to access multiple remote servers in one SQL request.

Oracle Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as variable character (A*n*V).

The following table lists data type mappings based on the value of VARCHAR:

Oracle Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*

Oracle Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
VARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR2 (*n*)	*n* is an integer between 1 and 4000	A*m*V	A*m*V	A*m*	A*m*

Syntax: **How to Control the Mapping of Variable-Length Data Types**

```
ENGINE SQLORA SET VARCHAR {ON|OFF}
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V). This is required for Unicode environments. ON is the default value.

OFF

Maps the Oracle data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A). ON is the default value.

Considerations for the CHAR and VARCHAR2 Data Types

Special attention must be paid to CHAR and VARCHAR2 data types. When you compare a CHAR data type column to a VARCHAR2 data type column, where the only difference is additional trailing spaces in the CHAR data type column, Oracle treats the column values as different.

The SET parameter lets you specify which of the two data types will be used for inserting, updating, and retrieving data.

If you create the tables outside of the server, we recommend that you use either CHAR or VARCHAR2 data types, but not both. If you create a table with both data types, you might not be able to retrieve the data you inserted due to Oracle's comparison mechanism. When inserting data into VARCHAR2 columns outside of the server, do not insert any trailing spaces.

If you use the server to generate Oracle tables and retrieve data, you will not encounter this problem, since the data type being used will be either CHAR or VARCHAR2, depending upon the ORACHAR setting.

Syntax: How to Set ORACHAR

```
ENGINE SQLORA SET ORACHAR {FIX|VAR}
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIX

Uses the CHAR data type. This is the default value.

VAR

Uses the VARCHAR2 data type.

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLORA SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Considerations for the NUMBER Data Type

When working with the NUMBER data type, where the precision is between 1 and 31, and scale is not negative and does not exceed precision, the data type is mapped to the server data type decimal (P) of corresponding precision and scale. Note that the resulting display length has to accommodate for sign and possible decimal point, thus it will exceed precision by 1 or 2.

When the NUMBER data type has precision 38 and scale 0, by default the data type is mapped to the server data type Integer (I), with a display length of 11.

Otherwise, by default, the NUMBER data type is mapped to the server data type double float (D), with a precision of 20 and a scale of 2.

Use the ORANUMBER setting to override the default mapping of the NUMBER data type, where the precision is between 32 and 38 and scale is not negative .

Syntax: How to Set ORANUMBER

```
ENGINE SQLORA SET ORANUMBER {COMPAT|DECIMAL}
```

where:

`SQLORA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE=SQLORA command.

`COMPAT`

Indicates that the NUMBER data type will be mapped using the default behavior described above.

`DECIMAL`

When NUMBER data type has precision between 32 and 38 and scale is not negative, the NUMBER data type precision (and possibly scale) will be truncated down to 31 prior to mapping, and then the data type will be mapped to the server data type decimal (P) as described above.

Reporting Against an Oracle Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Oracle stored procedures and report against procedure output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters (OUT parameters, and INOUT parameters in OUT mode), as well as the answer set. (Other methods of invocation retrieve the answer set only.)

- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 1792.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 1795.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

A synonym describes the parameters and answer set of a stored procedure.

An answer set's structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, "input parameters" refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception: if you know the procedure's internal logic, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

Reference: **Synonym Creation Parameters for Stored Procedures**

Parameter/Task	Description
Restrict Object Type to	Select *Stored Procedures*.
Filter by Owner/Schema and Object name	Selecting this option adds the Owner/Schema and Object Name parameters to the screen. **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end. **Object name.** Type a string for filtering the procedure names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all procedures whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.
Select	Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.
Name	The name of the synonym, which defaults to the stored procedure name.
Select Application	Select an application directory. The default value is baseapp.
Prefix/Suffix	If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters. If all procedures have unique names, leave the prefix and suffix fields blank.

Parameter/Task	Description
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
Customize data type mappings	To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed. For information about them, see *Oracle Data Type Support* on page 1788 manual.
Values	Select the check box for *every* parameter displayed for the specified procedure. Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines: ❏ Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided. ❏ Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym-creation purposes. If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.

Creating a Report Against a Stored Procedure

You can report against a stored procedure's answer set using the same facilities you use to report against a database table:

- **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 1796.
- **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 1795.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

`IF`

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

`*`

Indicates that you want to display all indicated parameters, or that there are no required parameters.

`WHERE`

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Oracle Environment

The Adapter for Oracle provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Choosing a Source for System Date and Time

When an SQL SELECT statement refers to CURRENT_DATE, CURRENT_TIME, or CURRENT_TIMESTAMP, it will obtain the date or time value from the Server (the default), or from the Oracle DBMS. You can override the default using the new SET DATETIME_PROCESS option. This is effective only for reports using Automatic Passthru.

Syntax: How to Choose a Source for System Date and Time

To choose a source for system date and time, the syntax is

```
SQL SET DATETIME_PROCESS=SERVER|DBMS
END
```

where:

SERVER

Specifies that the date and time values will be taken from the Server. This is the default.

DBMS

Specifies that the date and time values will be taken from the Oracle DBMS.

Designating a Default Tablespace

You can use the SET DBSPACE command to designate a default tablespace for tables you create. For the duration of the session, the adapter places these tables in the Oracle tablespace that you identify with the SET DBSPACE command. If the SET DBSPACE command is not used, Oracle uses the default tablespace for the connected user.

Syntax: How to Set DBSPACE

```
ENGINE SQLORA SET DBSPACE tablespace
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

tablespace

Is a valid tablespace in the database.

Note: This command will affect only CREATE FILE and HOLD FORMAT SQLORA requests issued by Table Services. It does not affect Passthru CREATE TABLE commands.

Overriding Default Parameters for Index Space

You can use the SET IXSPACE command to override the default parameters for the index space implicitly created by the CREATE FILE and HOLD FORMAT commands.

Syntax: How to Set IXSPACE

```
ENGINE SQLORA SET IXSPACE
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

index-spec

Is the portion of the Oracle CREATE INDEX statement that defines the parameters for the index. It can consist of up to 94 bytes of valid Oracle index space parameters. To reset the index space parameters to their default values, issue the SET IXSPACE command with no parameters.

Note: Refer to the Oracle documentation for more information on this command.

The long form of SQL Passthru syntax for commands exceeding one line is:

```
ENGINE SQLORA
SET IXSPACE index-spec
END
```

For example, to specify the NOSORT, NOLOGGING, and TABLESPACE portions of the Oracle CREATE INDEX statement, enter the following commands:

```
ENGINE SQLORA
SET IXSPACE NOSORT NOLOGGING
TABLESPACE TEMP
END
```

Note: This command will only affect CREATE INDEX requests issued by CREATE FILE and HOLD FORMAT SQLORA commands. It does not affect Passthru CREATE INDEX commands, for example:

```
ENGINE SQLORA SET IXSPACE TABLESPACE tablespace_name
TABLE FILE table_name
PRINT *
ON TABLE HOLD AS file_name FORMAT SQLORA
END
```

Activating NONBLOCK Mode

The Adapter for Oracle has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE SQLORA SET NONBLOCK {0|n}
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLORA SET PASSRECS {ON|OFF}
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying the Maximum Number of Parameters for Stored Procedures

You can use the SET SPMAXPRM command to specify the maximum number of input parameters that can be associated with any Oracle stored procedure.

Syntax: How to Set SPMAXPRM

```
ENGINE SQLORA SET SPMAXPRM nnn
```

where:

`SQLORA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nnn

Is the maximum number of parameters that can be passed to any stored procedure available to be run in this client session. 256 is the default value.

Oracle Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Optimizing Requests if a Virtual Field Contains Null Values

The SET OPTNOAGGR command provides finely-tuned control of adapter behavior for optimization. Users who for any reason wish to prevent passing aggregation to the RDBMS can use this command. An example of such a reason might be where NULL values occur in aggregated data with calculations. The SET OPTNOAGGR command causes the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is instead performed internally by the server while JOIN and SORT operations are handled by the RDBMS.

If any DEFINE field contains calculations with NULL fields then such operations cannot be translated to SQL and pass to DBMS because always return NULL. It has to be processed by FOCUS.

This can be achieved by SET OPTIMIZATION OFF.

However, in some cases it is preferable to use the off-load JOIN and SORT operation to DBMS for better performance while leaving AGGREGATION to FOCUS.

Syntax: How to Set Enhanced Aggregation Control

```
SQL SQLORA SET OPT {AGGR|NOAGGR}
```

where:

AGGR

Directs the adapter to off-load aggregated DEFINE fields to the DBMS. This is default setting.

NOAGGR

Directs the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is, instead, performed internally by the server, while JOIN and SORT operations are handled by the RDBMS. This setting can also be used to provide backwards compatibility for applications that were written based on the functionality of the previous release, when less SQL was off-loaded to the RDBMS. For example, when a calculation on aggregated fields may have contained NULL data that was not processed by the RDBMS NVL( ) function.

Example: Optimizing Aggregation Requests

The following example is based on the SCOTT.EMP SCOTT.DEPT demo Oracle tables.

```
JOIN CLEAR
JOIN DEPTNO IN EMP TO DEPTNO IN DEPT TAG D AS J1
DEFINE FILE EMP
LOC_FLAG/I2 = IF LOC EQ 'CHICAGO' THEN 1 ELSE 0 ;
BONUS_NEW/D15.2 = ((SAL + COMM)/2) * .2 ;
BONUS_BASE/D15.2 = (SAL + COMM) ;
END
TABLE FILE EMP
SUM COMM SAL BONUS_NEW MAX.LOC
FST.LOC_FLAG BONUS_BASE
-*
BY DNAME
BY ENAME
-*
ON TABLE HOLD
END
-RUN
TABLE FILE HOLD
PRINT *
END
```

With OPTIMIZATION ON, the default OPT AGGR adapter generates SQL, and if COMM is NULL, performs calculations in dbms and return 0 to report:

```
18.01.35 AE SELECT T2."DNAME",T1."ENAME", SUM(T1."COMM"), SUM(T1."SAL"),
18.01.35 AE SUM((((T1."SAL" + T1."COMM") / 2) * .2)), MAX(T2."LOC"),
18.01.35 AE MIN((CASE (T2."LOC") WHEN 'CHICAGO' THEN 1 ELSE 0 END)),
18.01.35 AE SUM((T1."SAL" + T1."COMM")) FROM EMP T1,DEPT T2 WHERE
18.01.35 AE (T2."DEPTNO" = T1."DEPTNO") GROUP BY T2."DNAME",T1."ENAME"
18.01.35 AE ORDER BY T2."DNAME",T1."ENAME";
```

This behavior can be overcome in two ways:

- Use SET OPTIMIZATION OFF to direct the adapter sent to dbms to use the most basic (atomic) SQL and put maximum processing on FOCUS.

  or

- Use SET OPT NOAGGR to pass JOIN to DBMS and leave aggregation to FOCUS, avoiding NULL calculations. Generated SQL still passes JOIN to Oracle to significantly improve performance:

```
18.02.36 AE SELECT T1."EMPNO",T1."ENAME",T1."SAL",T1."COMM",T1."DEPTNO",
18.02.36 AE T2."DEPTNO",T2."DNAME",T2."LOC" FROM EMP T1,DEPT T2 WHERE
18.02.36 AE (T2."DEPTNO" = T1."DEPTNO") ORDER BY T1."EMPNO",T2."DEPTNO";
```

Specifying Block Size for Retrieval Processing

The Adapter for Oracle supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify the Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE SQLORA SET FETCHSIZE n
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify the Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE SQLORA SET INSERTSIZE n
```

where:

SQLORA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Improving Efficiency With Aggregate Awareness

Aggregate awareness substantially improves the efficiency of queries.

For details about this feature, see *Aggregate Awareness Support* on page 2693.

Syntax: How to Set Aggregate Awareness

```
SET AGGREGATE_AWARENESS {FRESH_ONLY|OLD_OK|OFF}
```

where:

FRESH_ONLY

Sets different values for the parameters associated with each RDBMS.

OLD_OK

Sets different values for the parameters associated with each RDBMS.

OFF

If no option is selected, the behavior of the target RDBMS is determined by the database configuration options. There is no default for this setting.

For details about adapter-specific settings, see *Usage Notes for Aggregate Awareness* on page 2694.

Optimizing Non-equality WHERE-based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

LEFT_OUTER

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

hostfile

Is the host Master File.

AT

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

hfld1

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

tag1

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

WITH *hfld2*

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

MULTIPLE

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

UNIQUE

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations. It always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: Conditions for WHERE-Based Outer Join Optimization

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left-Outer Join

The following request creates a left outer conditional join between two Oracle data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE baseapp/EQUIP AT CARS
TO ALL FILE baseapp/CARREC AT CARC
WHERE CARS NE CARC;
END
TABLE FILE baseapp/EQUIP
PRINT CARS CARC STANDARD
BY MODEL
END
```

The WebFOCUS request is translated to a single Oracle SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."CARS"(CHAR( 16)),T1."STANDARD"(CHAR( 40)),
T2."CARC"(CHAR( 16)),T2."MODEL"(CHAR( 24)) FROM ( EQUIP T1 LEFT
OUTER JOIN CARREC T2 ON (T1."CARS" <> T2."CARC") ) ORDER BY
T2."MODEL";
```

Improving Optimizer Efficiency With Hints

DBMS Optimizer hints can be used to alter an execution plan. The adapter provides a setting which enable the TABLE command to place the hints after the SELECT keyword for Oracle.

This occurs when the adapter constructs a single SELECT statement. It does not occur in the case of a FOCUS-managed Join when multiple SELECTs are generated.

To reverse the setting, use SET HINT without the hint_text parameter.

Syntax: How to Setting Specific Hints

Use the following syntax to set specific hints:

```
SQL SQLORA SET HINT /* +hint_text */
```

Where

`SQLORA`

Is the target RDBMS. You can omit this value if you previously issued the SET SQLENGINE command.

`+hint_text`

Is the text of the hint or hints combination. Note that Oracle returns no error if the syntax is invalid. The Optimizer just ignores such hints. The end-user is responsible for the syntax. Omitting *+hint_text* resets the hint to none.

Example: Setting an Oracle Hints Combination

```
SQL SQLORA SET HINT /* +USE_HASH PARALLEL(EMPNO) INDEX_ASC */
TABLE FILE EMP
  PRINT EMPNO SAL BY DEPTNO
  IF DEPTNO GE 5
END
```

The WebFOCUS request is translated to a Oracle SELECT statement that incorporates the specified hints combination

```
SELECT
  /* +USE_HASH PARALLEL(EMPNO) INDEX_ASC */
  T1."EMPNO",
  T1."SAL",
  T1."DEPTNO"
FROM
  SCOTT.EMP T1
WHERE
  (T1."DEPTNO" >= 5)
ORDER BY
  T1."DEPTNO";
```

Calling an Oracle Stored Procedure Using SQL Passthru

Using SQL Passthru is supported for Oracle stored procedures. These procedures need to be developed within Oracle using the CREATE PROCEDURE command.

Note: Calling a stored procedure using SQL Passthru only allows the processing of one answer set per invocation. If multiple answer sets are expected, the CREATE SYNONYM mechanism is preferred.

Syntax: How to Invoke a Stored Procedure

```
SQL SQLORA EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

`SQLORA`

Is the ENGINE suffix for Oracle.

`procname`

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

parameter_specification

IN, OUT, and INOUT parameters are supported. Use the variation required by the stored procedure:

IN

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

OUT

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

INOUT

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

The adapter supports invocation of stored procedures that conform to the following rules:

- All scalar parameters (IN, OUT, and INOUT) are supported.
- A cursor must be defined with:
  - The TYPE statement in a PACKAGE or PROCEDURE.
  - An associated record layout of the answer set to be returned.
- The cursor must be opened in the procedure.
- No fetching is allowed in the stored procedure. The adapter fetches the answer set.
- Any messages must be issued using the RAISE APPLICATION ERROR method.
- If any parameters are declared as RECORD, and those parameters are associated with REFCURSOR, then when invoking the stored procedure using SQL Passthru do not specify those parameters. For an example, see *An Oracle Stored Procedure With REFCURSOR* on page 1811.

Any application error that is issued by the stored procedure is available in the server variable &ORAMSGTXT.

Example: An Oracle Stored Procedure With REFCURSOR

The following Oracle stored procedure uses REFCURSOR. You must create it in SQL*Plus using PL/SQL.

```
sqlplus scott/tiger
set serveroutput ON


   CREATE OR REPLACE PACKAGE pack1 AS
   TYPE nfrectype IS RECORD (
   employee NF29005.EMPLOYEE_ID5%TYPE,
   ssn5     NF29005.SSN5%TYPE,
   l_name   NF29005.LAST_NAME5%TYPE,
   f_name   NF29005.FIRST_NAME5%TYPE,
   birthday NF29005.BIRTHDATE5%TYPE,
   salary   NF29005.SALARY5%TYPE,
   joblevel NF29005.JOB_LEVEL5%TYPE);
   TYPE nfcurtype IS REF CURSOR RETURN nfrectype ;
   PROCEDURE proc1(c_saltable IN OUT nfcurtype);
   END pack1 ;
   /
   sho error


   CREATE OR REPLACE PACKAGE BODY pack1 AS
   PROCEDURE proc1 (c_saltable IN OUT nfcurtype)
   IS
   BEGIN
   OPEN c_saltable FOR SELECT EMPLOYEE_ID5,SSN5,LAST_NAME5,FIRST_NAME5,
   BIRTHDAT
   E5,SALARY5,JOB_LEVEL5 FROM NF29005;
   END proc1 ;  -- end of procedure
   END pack1; -- end of package body
   /
   sho error


/*
Invocation using SQL*Plus:
   VARIABLE c1 REFCURSOR
   EXEC scott.pack1.proc1 (:c1)
   PRINT c1

Invocation using SQL Passthru:
   SQL SQLORA EX scott.pack1.proc1
   TABLE FILE SQLOUT
   END
*/
```

Example: An Oracle Stored Procedure Without REFCURSOR

The following Oracle stored procedure does not use REFCURSOR. It includes IN and OUT parameters.

```
sqlplus scott/tiger
set serveroutput ON

   CREATE OR REPLACE PACKAGE pk1 AS
   PROCEDURE ibi_pr1 (v_num IN NUMBER,v_rec IN OUT VARCHAR2);
   END;
   /
   sho error

   CREATE OR REPLACE PACKAGE BODY pk1 AS
   PROCEDURE ibi_pr1 (v_num IN NUMBER,v_rec IN OUT VARCHAR2)
   IS
      v_msg    VARCHAR2(25);
      v_to_chr VARCHAR2(5);
   BEGIN
      v_to_chr := TO_CHAR(v_num);
      SELECT v_to_chr||' '||v_rec INTO v_msg FROM dual;
      v_rec := v_msg;
   END ibi_pr1;
   END pk1;
   /v
   sho error

/*
Invocation using SQL*Plus:
   VARIABLE v_rec VARCHAR2(20)
   -- assign IN value to IN OUT parameter:
   -- BEGIN :v_rec := 'Your message!...'; END;
   BEGIN :v_rec := &1; END;
   /
   EXEC scott.pk1.ibi_pr1 (&2,:v_rec)
   PRINT v_rec
   --


Invocation using SQL Passthru:
   SQL SQLORA EX scott.pk1.ibi_pr1 12345, ?/'AbCdEf';
   END
*/
```

Chapter 74 Using the Adapter for Oracle E-Business Suite

The Adapter for Oracle E-Business Suite is a read-only adapter providing security integration for reporting against Oracle E-Business Suite databases. This adapter supports user authentication and data access rules that are defined at the metadata and data access layer. Specific integrated data security is available for:

- Table and View Restrictions by Responsibility ID
- View Security by Set of Books ID
- View Security by Operating Unit

Using this adapter, you can access data with the following security models:

- Oracle Applications Security
- Union of Responsibilities Security
- Total Access Security

In this chapter:

Preparing the Oracle E-Business Suite Environment

Before adding an Oracle E-Business Suite, you must prepare the environment. The preparation steps are:

- Configure the Adapter for Oracle.
- Generate any required synonyms in the Adapter for Oracle.
- Customize the General Ledger Security Package.

Configuring the Adapter for Oracle

The Adapter for Oracle is used to support connectivity to the Oracle E-Business Suite.

To add a connection to the Oracle database, follow the instructions in *Configuring the Adapter for Oracle* on page 1773. When defining a connection to the Oracle E-Business Suite database, you must use the delivered database APPS user ID and the connection name must be VIS.

The APPS user ID is the database ID, not an application user ID. Privileges assigned to the APPS user ID enable the enforcement of data security rules when accessing data. Functions performed by this ID include authentication, setting the application user system context, data selection, and accessing views with row-level data restrictions.

Note: For this purpose, Information Builders recommends that the Reporting Server be run in PTH security mode.

Creating Oracle Metadata

The Adapter for Oracle E-Business Suite uses synonyms that you create using the Adapter for Oracle. For details about creating these synonyms, see *Managing Oracle Metadata* on page 1780.

Customizing the General Ledger Security Package

In order to execute views that contain calls to the Oracle E-Business Suite General Ledger Security Package (APPS.GL_SECURITY_PKG), the package must be customized to enable third party access. This package is called from within many General Ledger Business Views and performs the following functions:

- Updates temporary flex field tables.
- Uses the temporary tables in DDL of the views.

Procedure: How to Update the General Ledger Security Package

To update the General Ledger Package:

1. Use the Oracle Enterprise Management Console (or equivalent) to locate the package body GL_SECURITY_PKG found in the APPS schema.
2. Select *Edit*, *View* to modify this package body.
3. Locate the following syntax in the *init* procedure:

```
IF      (( instrb(program_name,'dis',-1,1) =  0 )      AND
  ( nvl(l_module,'XXX')          <> 'Discoverer4' )    AND
  ( nvl(l_gl_bis_disco_flag,'N')    <> 'Y' )           THEN
    return;
END IF;
```

4. Edit the syntax to include a new test on the program name, as follows:

```
IF      (( instrb(program_name,'dis',-1,1) =  0 )      OR
   ( instrb(program_name,'tsc',-1,1) =  0 ))          AND
   ( nvl(l_module,'XXX')         <> 'Discoverer4' )    AND
   ( nvl(l_gl_bis_disco_flag,'N')   <> 'Y' )           THEN
     return;
END IF;
```

This new syntax adds the tscom3.exe agent to the list of acceptable programs for the init procedure.

Data Access and Security

Oracle E-Business Suite relies on a combination of database functionality (views, stored procedures, and functions) along with interface rules to enforce business logic, such as data security rules. The adapter supports this design by enforcing security embedded rules within the database and by allowing queries to be written against both restricted and unrestricted types of data. Therefore, the adapter can be used to support data access with the following security models:

- **Oracle Applications Security.** Reports may be defined to require a Responsibility ID and Application ID as part of their processing. When this is done, requests against business views containing row-level security data restrictions will display the secured and filtered result data set only.
- **Union of Responsibilities.** When reporting against tables or views found in the applications, users will be able to access the full union of the data that all of their responsibilities allow. Only when Oracle Applications Security is defined will this data be further restricted to an individual responsibility ID access profile.
- **Total Enterprise Access.** Reports will always enforce table and view restrictions based on the application access granted to a given user. For all tables, and the subset of views that do not have row-level data security rules built-in, reports written against them will provide full data access.

Reference: Data Access and Security Limitations

- Synonyms created for use with this application adapter must be created with the same name as the underlying Oracle RDBMS Table or View.

- Oracle applications provide automated row-level security against many of the database views stored in its repository. These views are secured based on the initialization of environment settings during the user login process. The database tables do not support row-level security through this model automatically. To support row-level security against database tables, you must code filter criteria into individual reports.

Configuring the Adapter for Oracle E-Business Suite

The process of configuring the adapter includes configuring the adapter from either the Web Console or the Data Management Console, creating Oracle E-Business Suite control tables, and configuring the Remote Services Profile to call the adapter.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Procedure: How to Create the Adapter Control Tables

You can create the adapter control tables from either the Web Console or the Data Management Console.

1. From the Web Console, select *Applications*

   or

   from the Data Management Console, right-click the server and select *Connect*.

2. In the navigation pane, expand the *Application Directories* folder, right-click any application, and select *New*.
3. In the edit pane, type EX OESFILES and click the run icon. An example of the output follows:

```
0 NUMBER OF RECORDS IN TABLE=     2546  LINES=   2546

ORAUSER.FOC: Oracle eBusiness Suite User Info

...Loaded

0 NUMBER OF RECORDS IN TABLE=    42094  LINES=  42094

ORASYSF.FOC: Oracle eBusiness Suite Table Info

...Loaded

0 NUMBER OF RECORDS IN TABLE=    10290  LINES=  10290

ORAUSRAP.FOC: Oracle eBusiness Suite App Info

...Loaded
```

Procedure: How to Configure the Remote Services Profile

1. Edit or create the Remote Service profile for the client node to be used for Oracle E-Business Suite data access. To do so, open the WebFOCUS Administration Console, navigate to the Reporting Servers Node, and select *Profile* from the options at the bottom of the page.
2. Add the _site_profile call to the adapter procedure, OESLOGIN.

```
<VER1>

# Site Profile: Adapter Oracle E-Business Suite
# Description:  Enables remote user authentication against
#               the Oracle E-Business Suite and passes necessary
#               parameters to set the system context for the user
# Set default responsibility ID and application ID
#   Calling application must set and validate the resp ID
#   and appl ID before passing them to the procedure. If
#   these are not validated they will still be passed, but
#   invalid data may be returned for the user. If no values
#   are set, these defaults will be used in setting the system
#   context. In which case, certain database views will be
#   unusable.
# Bind MR User to WF User
<ifdef> IBIMR_user
IBIC_user=&IBIMR_user
<endif>
<ifdef> IBIMR_pass
IBIC_pass=&IBIMR_pass
<endif>
# Responsibility ID (IBIWF_respid)
<ifndef> IBIWF_respid
IBIWF_respid=0
<endif>
# Application ID (IBIWF_applid)
<ifndef> IBIWF_applid
IBIWF_applid=0
<endif>

# Execute parameterized call to OESLOGIN
_site_profile=&_site_profile\EX OESLOGIN OES_RESP_ID=&IBIWF_respid,
_site_profile=&_site_profile\n   OES_APPL_ID=&IBIWF_applid, PASSTHRU=ON
_site_profile=&_site_profile\n-RUN

# Enable these &variables to be passed to the server
<SET> IBIWF_respid(PASS)
<SET> IBIWF_applid(PASS)
```

Maintaining Security Rules

The OESSEC application contains FOCUS tables loaded with security settings at a specific point in time. In order to maintain current data access security rules within these FOCUS tables, you must rerun the OESFILES procedure periodically. This procedure can be rerun manually, on an as-needed basis.

Alternatively, you can schedule the execution of OESFILES on a regular basis. Sites that use ReportCaster for scheduling can execute OESFILES as a Server Procedure Schedule Task. Sites that do not use ReportCaster can use any standard operating system scheduling tool.

Chapter 75

Using the Adapter for Oracle TimesTen

The Adapter for Oracle TimesTen allows applications to access specific Oracle TimesTen data sources. The adapter converts application requests into Oracle TimesTen calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Oracle TimesTen Environment

In order to use the Adapter for Oracle TimesTen, you must install the Oracle TimesTen driver for whichever data source you would like to access, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the Oracle TimesTen Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some Oracle TimesTen Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and checking that the special services section displays "JSCOM3 active".

Configuring the Adapter for Oracle TimesTen

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Oracle TimesTen data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one Oracle TimesTen data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Oracle TimesTen

The *Oracle TimesTen* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Oracle TimesTen data source.

Driver name

Name for the Oracle TimesTen driver:

```
com.timesten.jdbc.TimesTenDriver
```

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLOTT SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLOTT`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the Oracle TimesTen data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the Oracle TimesTen Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLOTT SET CONNECTION_ATTRIBUTES CON1
  'jdbc:timesten:client:dsn=data_source_name'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLOTT SET DEFAULT_CONNECTION connection
```

where:

`SQLOTT`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLOTT SET DEFAULT_CONNECTION SAMPLE
```

Managing Oracle TimesTen Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLOTT to identify the Adapter for Oracle TimesTen.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLOTT [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLOTT`

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Oracle TimesTen

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 1831.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Oracle TimesTen synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLOTT ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Oracle TimesTen table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❏ KEYFLD is the FIELDNAME of the common column from the parent table. ❏ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support Report

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Oracle TimesTen Environment

The Adapter for Oracle TimesTen provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Oracle TimesTen.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLOTT SET TIMEOUT {nn|0}
```

where:

`SQLOTT`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native Oracle TimesTen driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLOTT SET PASSRECS {ON|OFF}
```

where:

`SQLOTT`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Oracle TimesTen Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 76

Using the Adapter for parAccel

The Adapter for parAccel allows applications to access specific parAccel databases. The adapter converts application requests into parAccel calls and returns optimized answer sets to the requesting application.

In this chapter:

Configuring the Adapter for parAccel

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Preparing the parAccel Environment

To use the Adapter for parAccel, you must first install the parAccel ODBC driver and all related libraries. To use this driver, you must configure a working data source using the ODBC Data Source Administrator. The Adapter will use this data source to access the parAccel database.

Declaring Connection Attributes

In order to connect to the parAccel database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf*), or in a group profile (if supported on your platform).

You can declare connections to more than one parAccel database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the parAccel server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for parAccel

The parAccel adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

This is the data source set in the ODBC Data Source Administrator.

Security

There are two methods by which a user can be authenticated when connecting to a data source:

- **Explicit.** The user ID and password are explicitly specified for each connection pass to the ODBC data source, at connection time, for authentication.
- **Trusted.** The adapter connects to the ODBC data source as a Windows login using the credentials of the operating system user impersonated by the server data access agent.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

File DSN parameters (optional)

Is the path to the ODBC File DSN that is defined on your Windows machine or network.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

For User/System DSN:

```
ENGINE SQLPARA SET CONNECTION_ATTRIBUTES connection
   DSNname/user_name,password,
```

For File DSN:

```
ENGINE SQLPARA SET CONNECTION_ATTRIBUTES connection
    FileDSN_name/,:'filedsn=FileDSN_path;',
```

where:

`SQLPARA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the parAccel Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`user_name`

Is the primary authorization ID by which you are known to an parAccel data source.

`password`

Is the password associated with the primary authorization ID.

`FileDSN_`*`name`*

Is the name of parAccel File Data Source Name (DSN) you wish to access.

`FileDSN_`*`path`*

Is the path to the parAccel File DSN, which may reside on the hard drive or on the network.

Example: Declaring Connection Attributes

For User/System DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the parAccel DSN named SAMPLESERVER.

```
ENGINE SQLPARA SET CONNECTION_ATTRIBUTES CON1
  SAMPLESERVER/SAMPLEUSER,SAMPLEPASSWORD,
```

For File DSN:

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 via the ODBC File DSN named SAMPLESERVER.dsn.

```
ENGINE SQLPARA SET CONNECTION_ATTRIBUTES CON2
SAMPLESERVER.dsn /,:'filedsn=C:\ SAMPLESERVER.dsn;'
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLPARA SET DEFAULT_CONNECTION connection
```

where:

`SQLPARA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLPARA SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLPARA SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLPARA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing parAccel Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the parAccel data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLPARA to identify the Adapter for parAccel.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLPARA [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLPARA

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each parAccel table or sheet that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for parAccel

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Note: Due to internal settings of the parAccel, filtering by Owner/Schema might not be applicable to some database objects.

Location of External SQL Scripts

If you specify External SQL Scripts in the Restrict Object type to field, these additional fields are displayed. During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 1847.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for parAccel synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File

```
FILENAME=ONHAND, SUFFIX=SQLPARA, $
  SEGMENT=ONHAND, SEGTYPE=S0, $
    FIELDNAME=PROD_NUM, ALIAS=Prod_Num, USAGE=A255V, ACTUAL=A255V,
      MISSING=ON, $
    FIELDNAME=PRODNAME, ALIAS=Prodname, USAGE=A255V, ACTUAL=A255V,
      MISSING=ON, $
    FIELDNAME=QTY_IN_STOCK, ALIAS=Qty_in_Stock, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=PRICE, ALIAS=Price, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
    FIELDNAME=COST, ALIAS=Cost, USAGE=D20.2, ACTUAL=D8,
      MISSING=ON, $
```

Access File

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DB1,KEYS=1,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File for parAccel.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the parAccel table. The value assigned to this attribute can include the name of the workbook as follows: TABLENAME=[*filename*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.

Keyword	Description
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Chapter 77

Using the Adapter for PeopleSoft

The Adapter for PeopleSoft provides data access for reporting from the following PeopleSoft Record Types: SQL tables, SQL views, and query views. The adapter fully enforces PeopleSoft Query tree and row-level security.

The adapter provides three types of services:

- Authentication services.
- PeopleSoft data source connection management services.
- Metadata administration and data access security services.

In this chapter:

Preparing the PeopleSoft Environment

Prior to configuring the Adapter for PeopleSoft, you must ensure that minimal software requirements have been met. In addition, you must define a PeopleSoft Access ID to enforce PeopleSoft security. Administrators must also review available connection options before beginning the initial configuration.

Software Requirements

The following software is required on the server to support the Adapter for PeopleSoft:

- An appropriate RDBMS connectivity product on the computer where you will install the Adapter for PeopleSoft server. This is to provide access to a PeopleSoft data source server for operational data access.
- To support integrated authentication to PeopleSoft 8.1 or higher, Java 2 SDK version 1.3.1 or higher must be installed. Additionally, Publish and Subscribe services must be active on the PeopleSoft Application server.

Note: For information about PeopleSoft with versions of software other than those listed in this topic, contact your Information Builders representative.

Setting Up the PeopleSoft Access ID

The Adapter for PeopleSoft initially authenticates with PeopleSoft using a PeopleSoft application user ID and password. If successful, the adapter attaches to the RDBMS using an access ID of your choice. Typically, this access ID is either the default PeopleSoft access ID (often it is SYSADM), or it is an RDBMS ID dedicated to the adapter. In either case, the access ID must have read access to the following PeopleSoft tables and views:

PSCLASSDEFN	PSDBFIELD	PSDBFLDLABL
PSOPRCLS	PSOPRDEFN	PSOPTIONS
PSRECDEFN	PSRECFIELD	PSRECFIELDDB
PS_SCRTY_ACC_GRP	PS_TREE_ACCESS_VW	PSTREEDEFN
PSTREENODE		

In addition, the access ID must have read access to any PeopleSoft Records that you wish to report against. If your PeopleSoft Records have row-level security, this ID must also have read access to the Query Security Records associated with each.

Configuring the Adapter for PeopleSoft

In order to configure PeopleSoft, you must create and register the PeopleSoft application (SNAPCAT) in the catalog path, add the underlying SQL adapter, add the first PeopleSoft connection, and, optionally, configure a JSCOM3 listener to ensure password authentication for PeopleSoft 8 and higher.

Creating the SNAPCAT Application

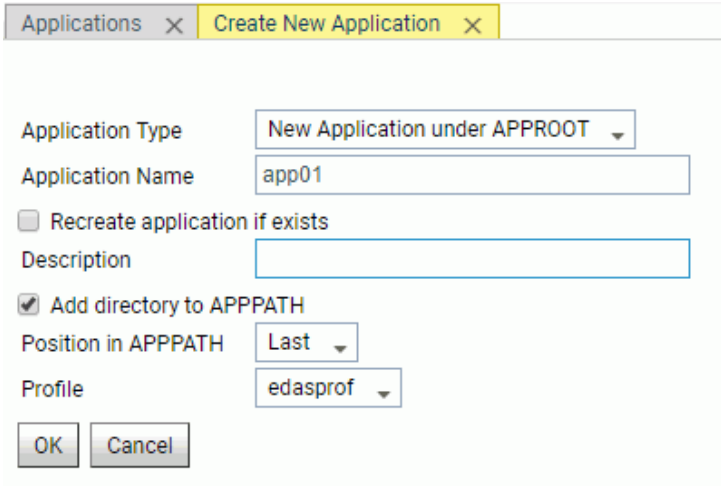
Before you can configure the PeopleSoft adapter, the SNAPCAT application must be created and added to the application path of the server. This can be done through the Web Console.

Note: In the server release 7.7.04 and higher, it is no longer necessary to include the SNAPINST application directory in the application PATH. This directory is automatically added or removed from the PATH when needed.

Procedure: How to Create the SNAPCAT Application

1. From the Web Console menu bar, select *Applications*.
2. Right-click the *Application Directories* folder and select *New*, *Application Directory*.

   The Create New Application window opens, as shown in the following image.



3. Change the Application Name to *snapcat*. Make sure it is all in lowercase.
4. Click *OK*.

Adding the SQL Database Adapter

The Adapter for PeopleSoft is an enhanced version of the underlying data source adapter. The adapter must be added into the server configuration.

When you add an SQL adapter, there is no need to provide configuration parameters since these connection parameters are stored in the server profile, which is accessible at all times to any user or application connecting to the server.

For most applications, the data access is then secured through procedure logic and operating system security. With PeopleSoft, this is not possible as there may be very large numbers of users who do not have operating system login IDs. Instead, these configuration parameters are supplied later when configuring connections to PeopleSoft data sources.

As a recommended practice when adding the new adapter, you may initially supply the configuration parameters, test data source connectivity, and then delete the connection statement in the server profile. While not required, when you attempt to create a connection to the PeopleSoft data source, this process may save significant time.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Configuring PeopleSoft Password Authentication

To leverage integrated authentication against PeopleSoft, a JSCOM3 listener must be configured using the Web Console. When completed, all user requests will pass calls to the PeopleSoft Component Interface architecture for authentication. If the configuration is either not using PeopleSoft or the connections are to be configured with TRUSTED authentication, then this step is not necessary.

Note: At this time, the PeopleSoft Component Interface does not support LDAP, even though the PeopleSoft Internet Architecture does read from LDAP repositories in general. This limitation means that if your PeopleSoft site uses LDAP integration for user authentication only, TRUSTED authentication is possible. To leverage your LDAP repository for WebFOCUS, use any of the standard integration methods available. For details, see the *WebFOCUS Security and Administration* manual.

Before configuring JSCOM3, the following conditions must be met:

- Ensure that the Publish and Subscribe services are running on the PeopleSoft Application Server. By turning these on, the jolt listener becomes active. Usually, these services are used for application messaging integration requirements, but the Adapter for PeopleSoft uses the same technology for authentication. No particular component interface must be accessible for the user to authenticate.
- Several API files delivered with PeopleSoft must be made available to the Reporting Server directory. These files are located in the PeopleSoft file server libraries in the following locations:

  `[PeopleSoft file server]\web\PSJOA\psjoa.jar`

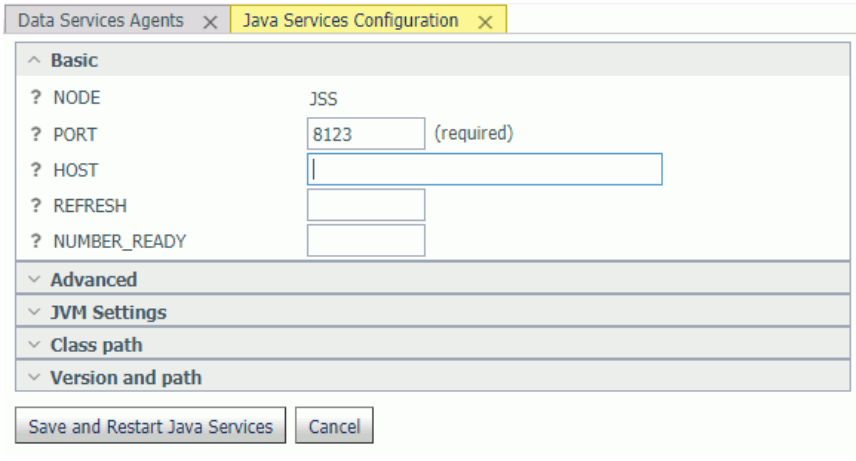
If the Reporting Server is installed on the same computer as your PeopleSoft Application Server, then you may configure JSCOM3 to point directly to the files. Otherwise, the files must be copied to the computer with the Reporting Server. In this situation, it is recommended (but not required) that they be placed in your EDACONF\bin directory.

Procedure: How to Configure the JSCOM3 Listener

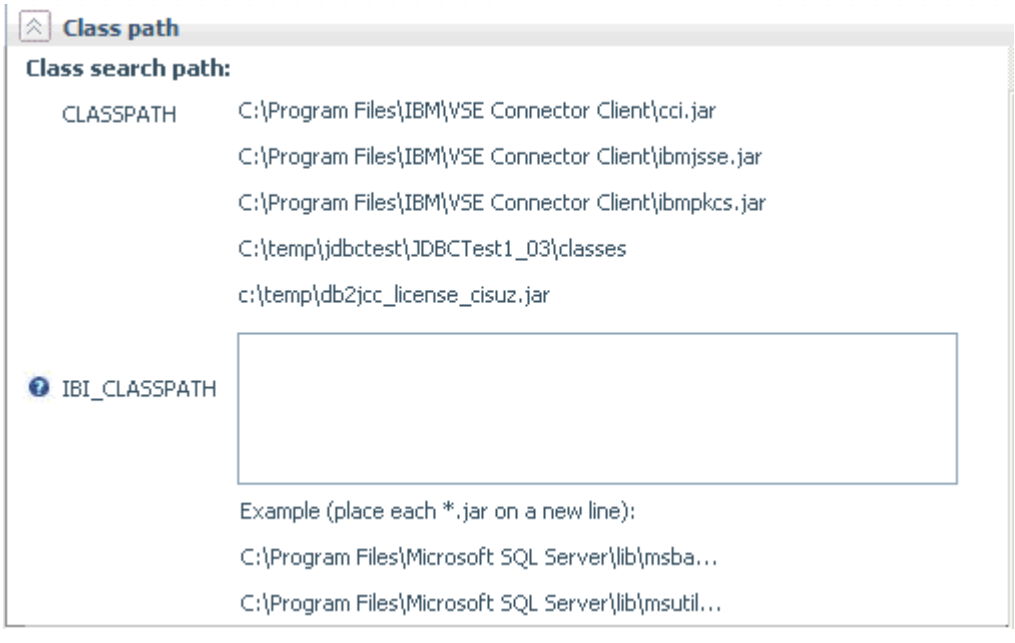
To configure the JSCOM3 listener:

1. From the Web Console menu bar, select *Workspace*.
2. From the navigation pane, expand the *Java Services* folder.
3. Right-click *DEFAULT*, and select *Properties*.

The Java Services Configuration pane opens.



4. Expand the *Class path* arrow.



5. Type the location of the psjoa.jar file in the IBI_CLASSPATH field, including the file name, for example, C:\Program Files\PeopleSoft\lib\psjoa.jar

Note: When upgrading the PeopleSoft Application Server after having performed these configuration steps, you must update the PeopleSoft files (psjoa.jar).

Adding the First Connection to PeopleSoft

Although multiple PeopleSoft connections can be created, the first connection requires slightly different management steps. The reason for this is that before any connections to PeopleSoft data sources can occur, a base layer of application metadata is created and configured. This occurs transparently for the Administrator, but is a slightly different connection management process than when adding additional connections.

To create the first connection:

- ❑ Select and type a DBA password.

  The DBA password is the value that is used as a metadata access key. To access the Web Console, the administrator must know this value. Users outside of the Web Console do not have metadata access without first logging in to PeopleSoft through this integration.

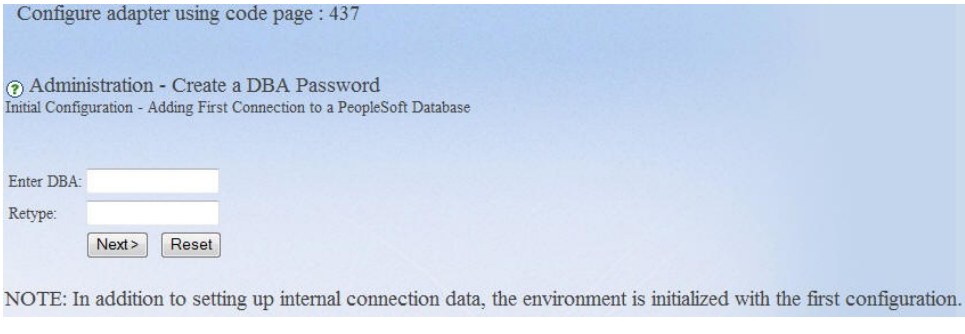
- ❑ Add a PeopleSoft Connection.

Note: In server release 7.7.04 and higher, it is no longer necessary to include PSLOGIN as a service profile.

Procedure: How to Add the First PeopleSoft Connection

1. Open the Web Console.
2. From Web Console menu bar, click *Adapters*.
3. Expand the *Available* folder, if it is not already expanded.
4. Expand the *ERP* group folder.
5. Right-click *Peoplesoft*, and select *Configure*.
6. Click *Configure*.
7. Click *Next*.

   The Administration - Create a DBA Password window opens.



8. In the *Enter DBA and Retype* fields, specify a DBA of up to eight characters, then click *Next*.

   The Administration - Create a New PeopleSoft Connection window opens.

9. Type the necessary information, test the connection, and if successful, click *Configure*. If not successful, edit the information until the test is successful, then click *Configure*. For more information, see step 5 in *How to Add a PeopleSoft Data Source Instance* on page 1881.

   Note: The *Configure* button becomes available once the connection test to the selected RDBMS is successful.

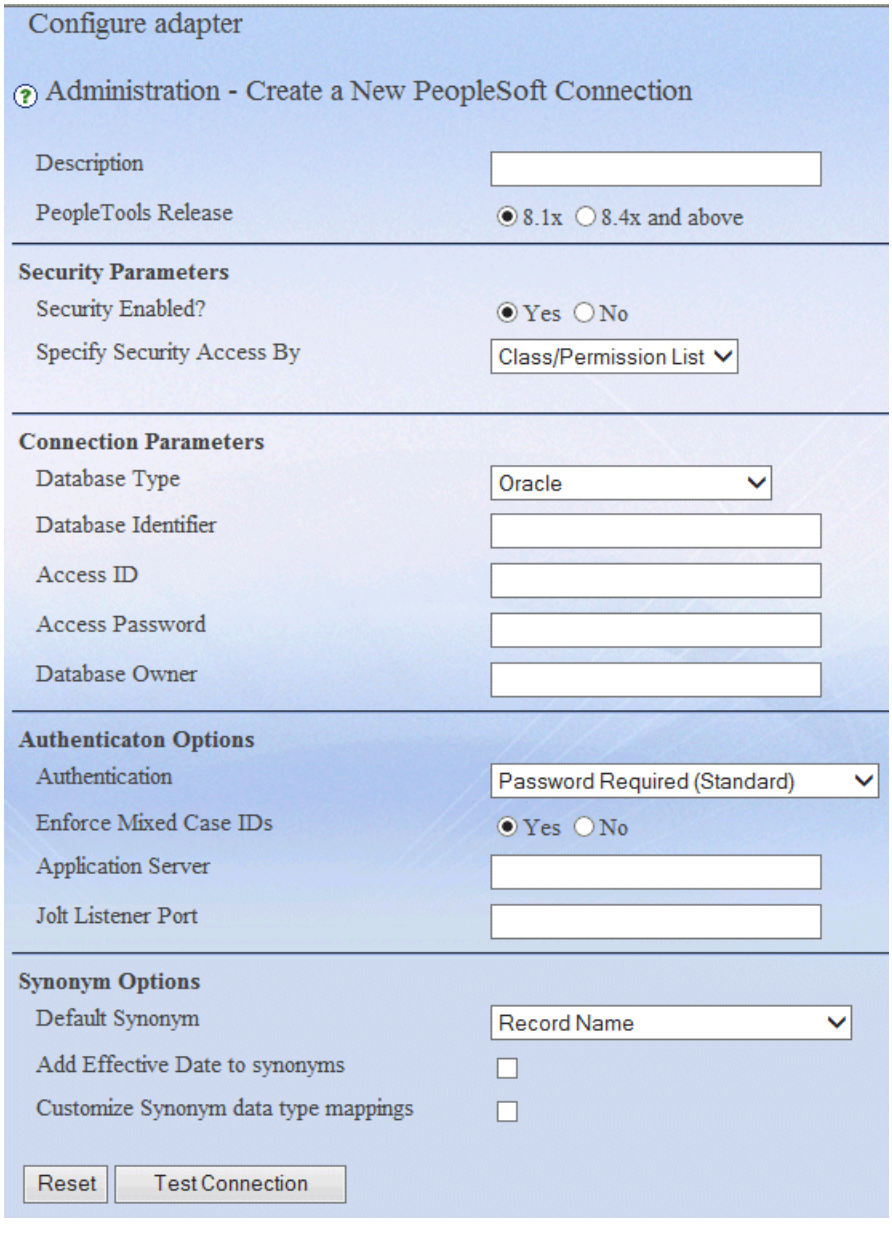
10. Once the configuration finishes, click *Next*.

    Most of the parameters can be changed in the future, but the connection cannot be created without initially supplying correct data source connectivity information. For details on the required information, see *Administration - Create a New PeopleSoft Connection Window* on page 1857.

    Tip: A worksheet is provided to assist you in gathering the information you will need to type on this screen. For details, see *PeopleSoft Connection Worksheet* on page 1862.

    Processing may take several minutes to complete. Once completed, you may begin managing your new PeopleSoft Connection by adding synonyms and administering security access.

Reference: **Administration - Create a New PeopleSoft Connection Window**



The Administration - Create a New PeopleSoft Connection window contains the following fields/ options:

Description

Is a description that identifies the current server configuration.

PeopleTools Release

Is the major PeopleSoft PeopleTools release number to access.

Security Parameters

Security Enabled?

Determines whether to enforce PeopleSoft data access security. The options are Yes and No. Yes is the default value.

Specify Security Access By

Determines the way in which the PeopleSoft administrator enables users to access the adapter and PeopleSoft data. The options are:

- **Class/Permission List.** Manages users in operating class (permission list) grouping.
- **User/Operation ID.** Manages users by individual user IDs.

Connection Parameters

Database Type

Is the Data source type. The options are: Oracle, MS SQL Server, and Db2 Universal Database.

Database Identifier

Is the data source identifier or system data source name.

If the server is co-located with the data residing in Oracle, do not type anything in the Database Identifier field.

Server

Name of the Database Server (for SQL Server only, not Oracle or Db2). If the instance of SQL Server you are configuring for is not running with the default port number, then the following syntax should be used:

nnn.nnn.nnn.nnn\instance_name, port

where

nnn.nnn.nnn.nnn

Is the IP address of the machine where SQL Server is running.

instance_name

Is the name of the SQL Server instance.

port

Is the listening port of the SQL Server instance.

Access ID

Is the RDBMS login ID for data source connectivity. Typically, it is the same login ID PeopleSoft uses to access the data source. There are minimum data access requirements that the ID must have. For details, see *Setting Up the PeopleSoft Access ID* on page 1850.

Access Password

Is a password associated with the access ID.

Database Owner

Is the data source owner identifier or the schema owner depending on your data source. The owner ID is used to fully qualify SQL requests passed to the data source. This allows multiple data source instances or schemas to exist in a single data source. Typical defaults are SYSADM on Oracle and dbo on Microsoft SQL Server.

Authentication Options

Authentication

Determines the authentication used in your application. The options are:

- **TRUSTED (Alternate Authentication).** Expects a valid pre-authenticated PeopleSoft user ID to be passed to the server to enable data access. This is useful with LDAP, Operating System security, or some other application that authenticates the user.
- **Password Required (Standard).** Expects a valid user ID and password before allowing data access. This is the default and is recommended for most environments.

Enforce Mixed Case IDs

Determines whether passwords are case-sensitive. Select *Yes* to enforce the PeopleSoft 8 use of mixed case user IDs or select *No* to revert to PeopleSoft 7.x case insensitivity.

If Password Required (Standard) is selected, two additional input fields are displayed:

Application Server

This is the IP address of the machine on which the JOLT Listener resides for the configured PeopleSoft environment.

JOLT Listener Port

Is the port number of the JOLT Listener.

Synonym Options

Default Synonym

Is the type of synonym to default to when creating new metadata. The options are: Record Name, Record Name with Prefix, Record Name with Suffix, Table Name, Table Name with Prefix, Table Name with Suffix.

Note: If you specify any of the options with Prefix or Suffix, the following additional input box is displayed.

Prefix/Suffix Default

Enter the prefix or suffix you wish to prepend or append to the synonym name.

Add Effective Date to synonyms

If this option is selected, additional field entries are added to any synonym created (or refreshed) after the configuration is saved. These new field entries provide the End Date of any row where an effective date field is part of the key. If an effective sequence is also part of the key, then the effective sequence field is also used in the calculation for the *End Date* field. In addition, filters are added to the synonym for the following:

- Current Record

and, if *Effective Sequence* field is present, the following filters are also added:

- First Effective Sequence
- Last Effective Sequence
- All Effective Sequences

These filters can be used in requests against the synonym and are available in the WebFOCUS reporting tool set.

To take advantage of this feature when migrating from an earlier release to release 7.7.02, first following the instructions in *Migrating from 7.1x and 7.6.x to 7.7* on page 1884, then edit each connection that needs this functionality, select the new option, and save the configuration. All synonyms should now be refreshed to have the new fields/filters added to them.

Remove Data Mask Fields

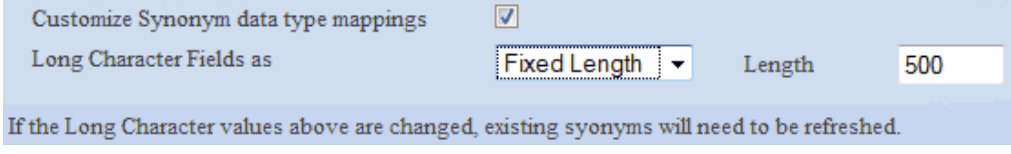
This option will remove any masking fields from the row level security segment of the generated synonym. These fields are identified by having _MSK at the end of the field name. The masking field along with the masked field will not be present in the synonym if this option is checked. This is for version 9 Campus Solutions Customers who have implemented masking using SACR.

Customize Synonym data type mapping

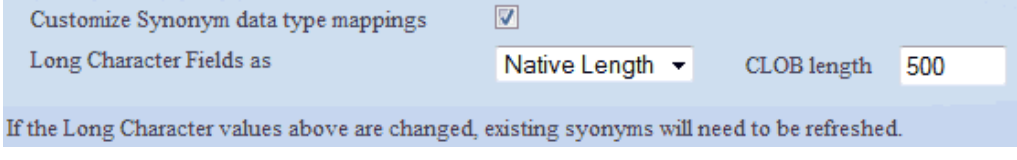
This option specifies how Long Character and CLOB data types in the PeopleSoft table should be reflected in the synonym.

Long Character Fields as

Fixed Length – Any Long Character or CLOB will be converted to the Fixed Length value. In the following image, the synonym will have A500V for these two data types.



Native Length – Long Character fields will reflect the length as specified in the PeopleSoft table definition, CLOB will be converted to the Fixed Length value. In the following image, the synonym will have A500V for the CLOB data types.



Length or CLOB Length

Length – This is the length that will be used for both Long Character and CLOB fields if *Fixed Length* is selected.

CLOB Length – This is the length that will be used for all CLOB fields if *Native Length* is selected.

If this is a new connection configuration, all synonyms created after the configuration is saved will have these values. If an existing connection is being modified, synonyms will have to be refreshed to use these values.

Reference: PeopleSoft Connection Worksheet

Use the following worksheet to record connection configuration information and keep it for future reference.

Option Name	Response	Description
Description		Row.
DBA		8-character identifier for linking metadata to PeopleSoft security rules. Must be consistent across multiple connections.
PeopleTools Release		Major release of PeopleTools (8.1x, or 8.4x and above)
Security Enabled?		Specifies whether PeopleSoft data security (row-level and access group) is enforced.
Specify Security Access By		Specifies security access; by User (OPRID) or permission list (OPRCLASS).
Database Type		Designation for RDBMS. The options are Oracle, MS SQL Server, or Db2 Universal Database.
Database Identifier		Database System Identifier (SID) or Source Name used by the appropriate DBMS client software on the server.

Option Name	Response	Description
Server		Name of the Database Server (for SQL Server only, not Oracle or Db2). If the instance of SQL Server you are configuring for is not running with the default port number, then the following syntax should be used: *nnn.nnn.nnn.nnn\instance_name, port* where *nnn.nnn.nnn.nnn* Is the IP address of the machine where SQL Server is running. *instance_name* Is the name of the SQL Server instance. *port* Is the listening port of the SQL Server instance.
Access ID		ID used by PeopleSoft to connect to the RDBMS.
Access Password		Password associated with the access ID.
Database Owner		Data source owner ID for the PeopleSoft data source.
Authentication		Select *Password* to always verify the user ID/password. Select *Trusted* to accept only user ID. Trusted authentication is useful when deploying in previously authenticated environments, such as a Portal.
Enforce Mixed Case IDs		Provides backwards compatibility for sites that previously used a case-insensitive version of PeopleSoft. (Yes or No)
Application Server		This is the IP address of the machine on which the JOLT Listener resides for the configured PeopleSoft environment.
JOLT Listener Port		Is the port number of the JOLT Listener.

Option Name	Response	Description
Default Synonym		The type of synonym to default to when creating new metadata. ❏ Record Name ❏ Record Name with Prefix ❏ Record Name with Suffix ❏ Table Name ❏ Table Name with Prefix ❏ Table Name with Suffix **Note:** If you specify any of the options with Prefix or Suffix, an additional input box is displayed for your Prefix or Suffix entry.
Prefix/Suffix Default		Enter the prefix or suffix you wish to prepend or append to the synonym name.
Add Effective Date to synonyms		If this option is selected, additional field entries are added to any synonym created (or refreshed) after the configuration is saved. These new field entries provide the End Date of any row where an effective date field is part of the key. If an effective sequence is also part of the key, then the effective sequence field is also used in the calculation for the *End Date* field. In addition, filters are added to the synonym which can be used in requests against the synonym and are available in the WebFOCUS reporting tool set.

Option Name	Response	Description
Add Remove Data Mask Field		This option will remove any masking fields from the row level security segment of the generated synonym. These fields are identified by having _MSK at the end of the field name. The masking field along with the masked field will not be present in the synonym if this option is checked. This is for version 9 Campus Solutions Customers who have implemented masking using SACR.

Managing PeopleSoft Metadata

The Adapter for PeopleSoft enables management of PeopleSoft metadata. Processes exist that manage the accessibility of PeopleSoft Records by maintaining the metadata catalog on the reporting server. Before reports can be created or run, this metadata catalog must contain information about the PeopleSoft Record Definitions.

You can perform the following tasks to accomplish this:

- ❑ Create synonyms for PeopleSoft Records.
- ❑ Remove synonyms.
- ❑ Update synonyms.

Note: Only one administrator can access the adapter at any given time.

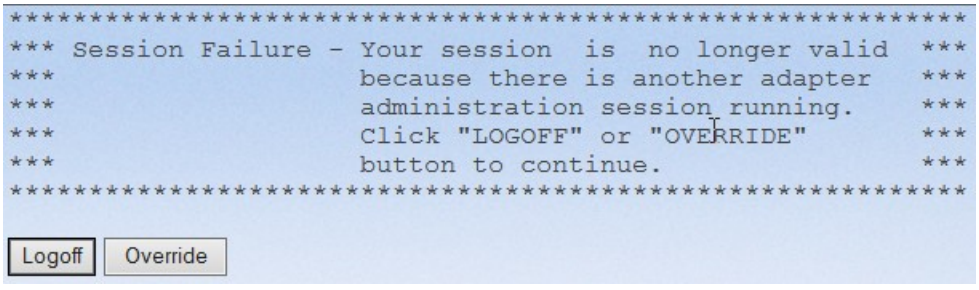
Accessing the Adapter for PeopleSoft Administrator

To access the environment for administration, you must have access to the Web Console and knowledge of the DBA.

Procedure: How to Access the Adapter for PeopleSoft Administrator

1. Open the Web Console.
2. From Web Console menu bar, click *Adapters*.
3. In the navigation pane, right-click *PeopleSoft* from the list of configured adapters, and select *Properties*.

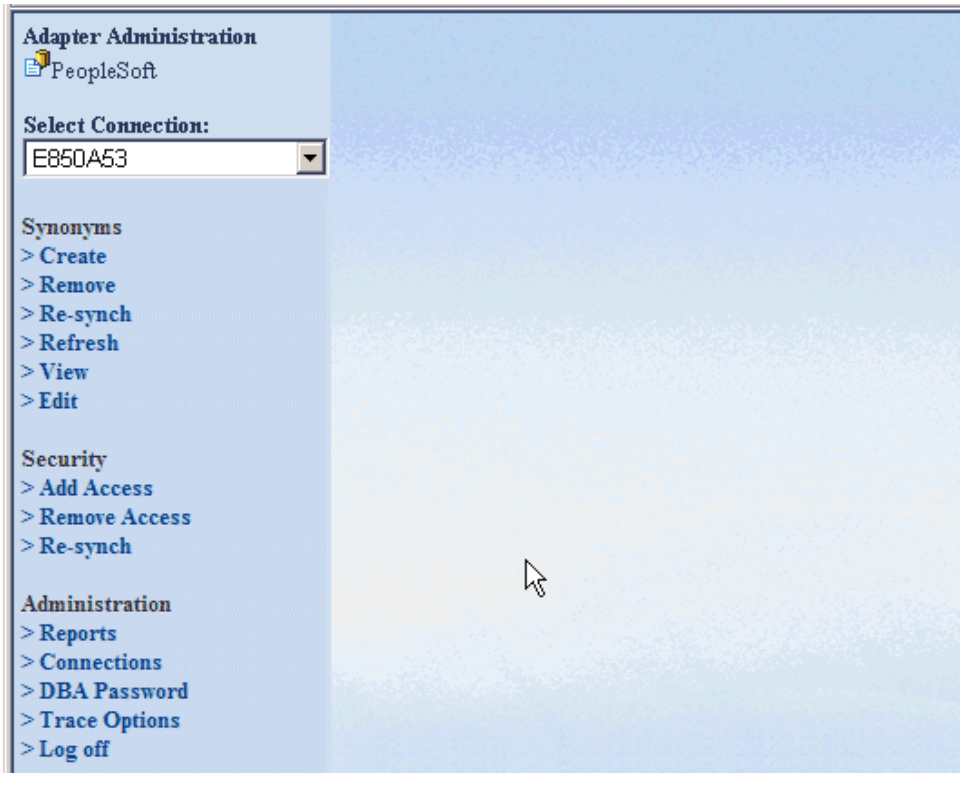
At this point, the system will check to see if any other administrator is actively using the adapter. If so, the following screen will be displayed.



If there is no other legitimate administrator session, click the *Override* button. Otherwise, click the *Logoff* button to end your adapter session. If you select to override the other session, that session will now become invalid.

4. Type the DBA password, and click *Next*.

The PeopleSoft menu opens.



Creating Synonyms

Creating synonyms for PeopleSoft is one of the core functions of the PeopleSoft Administrator. The adapter enables the administrator to choose which PeopleSoft records are defined in the metadata catalog.

The adapter provides two methods to select PeopleSoft Records to create synonyms:

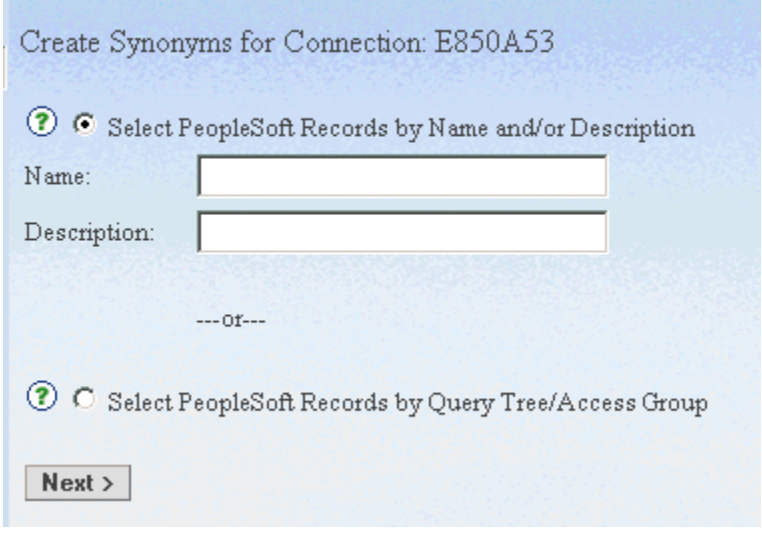
- Create synonyms using a filtered search. A search template option is provided to speed the process of searching for specific PeopleSoft Records. This is a useful method of locating Record Definitions when either the query tree location is not known or specific definitions that meet a filter can be more readily accessed.

- Create synonyms using a query tree search. The query tree is a hierarchical logical grouping of the PeopleSoft Record Definitions. This method provides administrators familiar with this organizational structure a way to quickly locate related record definitions. Many sites create a combination query tree and access group definition specifically to group their records for use in reporting.

Procedure: How to Create Synonyms Using a Filtered Search

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Create* under the *Synonyms* group.

   The Create Synonyms for Connection window opens.



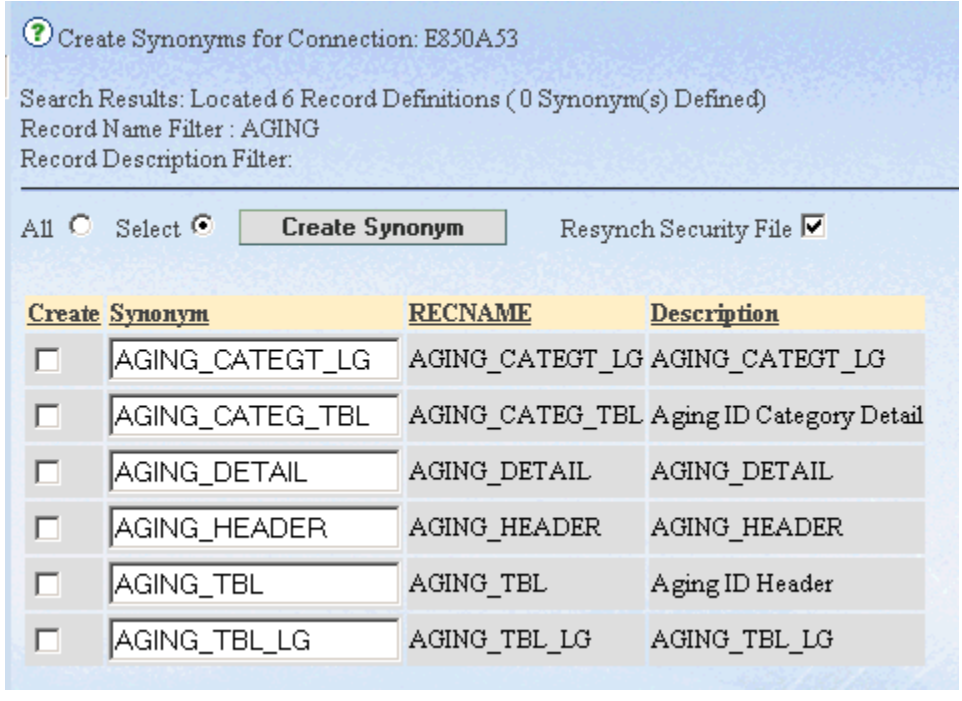
3. Click *Select PeopleSoft Records by Record Name and/or Description* and type a Record name or description to filter by in the respective fields, or leave the fields blank to see a full list of Records.

   Note:

   - The Record Description is case-sensitive.
   - If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

4. Click *Next*.

The Create Synonyms for Connection window opens.



5. Select the check box in the *Create* column for all the Records you want to create a synonym for, or select *All* to check all available Records.
6. Optionally, change the synonym name in the *Synonym* name column.
7. Uncheck the *Resynch Security File* check box if you have more synonyms to create during this session. Leave it checked for the last Create Synonym run.
8. Click *Create Synonym* to process. Processing may take a few minutes for each selected Record Definition.

Procedure: How to Create Synonyms Using a Query Tree Search

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Create* under the *Synonyms* group. The Create Synonyms for Connection window opens.
3. Click *Select PeopleSoft Records by Query Tree/Access Group* and click *Next*.

The Create Synonyms - Select Query Tree for Connection window opens.

Create Synonyms - Select Query Tree for Connection: E850A53

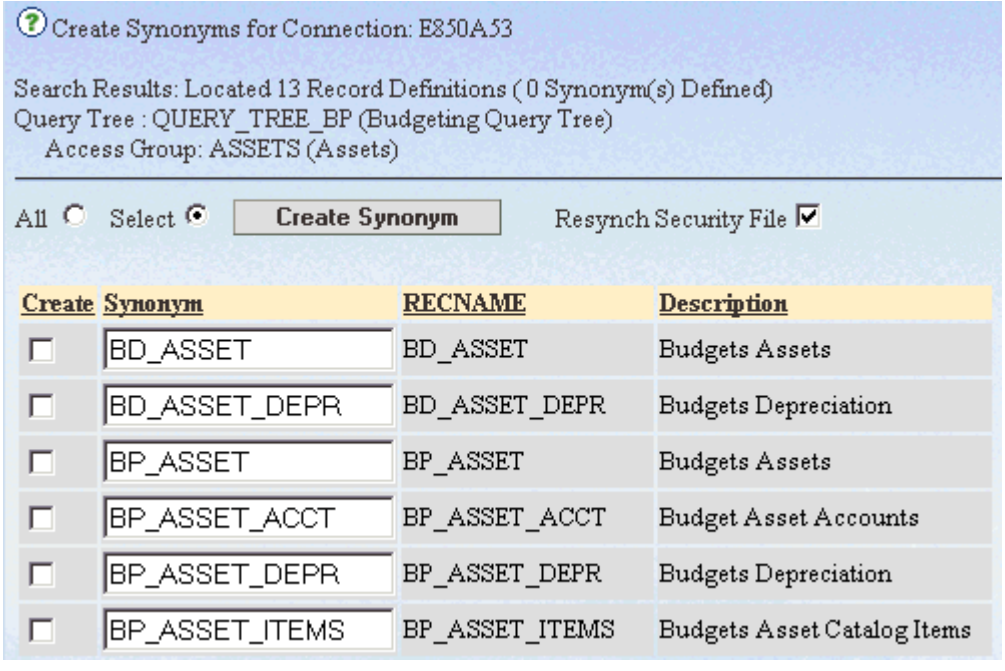
Query Tree	Description
PACKAGING	EM Product License Packaging
QUERY_TREE_AB	ABM Access Group
QUERY_TREE_BP	Budgeting Query Tree
QUERY_TREE_BSC	BSC Query Tree
QUERY_TREE_DP	demand planning query tree
QUERY_TREE_EO	Enterprise Components
QUERY_TREE_EOCM	Catalog Management
QUERY_TREE_EW	Query Tree for EW
QUERY_TREE_FI	FSI Access Group
QUERY_TREE_FT	FTP Access Group
QUERY_TREE_GC	Consolidation Query Tree
QUERY_TREE_IP	Inventory Policy Query Tree
QUERY_TREE_KPI	KPI Quey Tree
QUERY_TREE_ODS	ODS Reporting tables
QUERY_TREE_OLAP	Cube Manager generated records
QUERY_TREE_PF	ABC_QUERY
QUERY_TREE_PT	PeopleTools Query Tree
QUERY_TREE_RW	RWC Access Group
QUERY_TREE_WA	
QUERY_TREE_WF	Workflow Query Tree
QUERY_TREE_WFP	Workforce Planning Access Grp
UPG_QUERY_TREE	EPM Upgrade Query Tree

4. Select a Query Tree from the table by clicking the hyperlinked name.

The Create Synonyms - Select Access Group for Connection window reflects your selection.

5. In the Create Synonyms - Select Access Group for Connection window, click a hyperlinked name to drill down on an access group.

The Create Synonyms for Connection window opens.



6. Select the check box in the *Create* column for all the records you want to create a synonym for, or select *All* to check all available records.
7. Optionally, change the synonym name in the *Synonym* name column.
8. Uncheck the *Resynch Security File* check box if you have more synonyms to create during this session. Leave it checked for the last Create Synonym run.
9. Click *Create Synonym* to process. Processing may take a few minutes for each selected record definition.

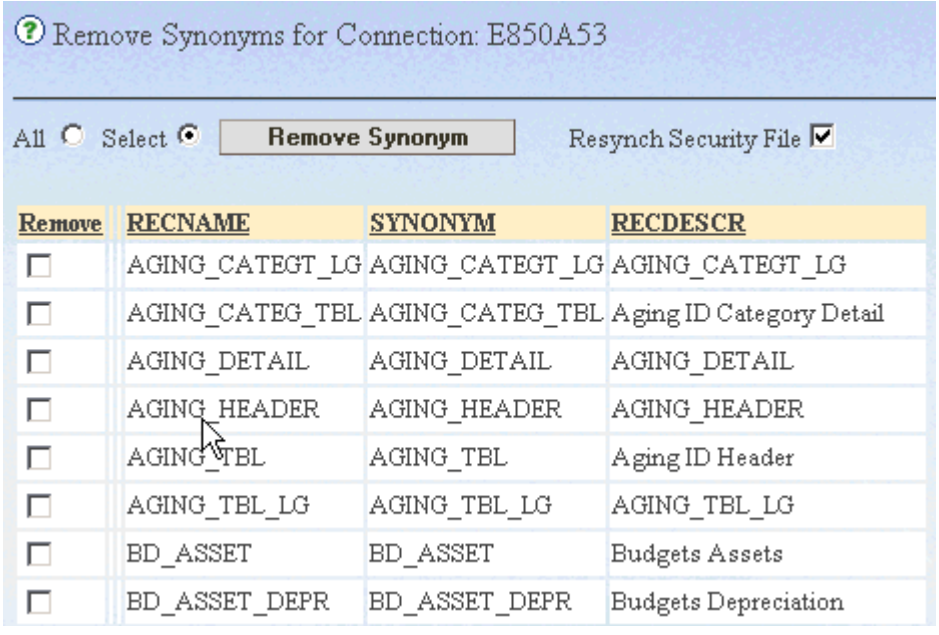
Removing Synonyms

You can remove previously created PeopleSoft synonyms from the metadata catalog with the Remove Synonyms menu option.

Procedure: How to Remove Synonyms

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Remove* under the Synonyms group.

The Remove Synonyms for Connection window opens.



3. Select the check box in the *Remove* column for each synonym you want to remove, or select *All* to check all available synonyms.
4. Uncheck the *Resynch Security File* check box if you have more synonyms to remove during this session. Leave it checked for the last Remove Synonym run.
5. Click *Remove Synonym*.

Updating Synonyms

You can re-synchronize your metadata. This option:

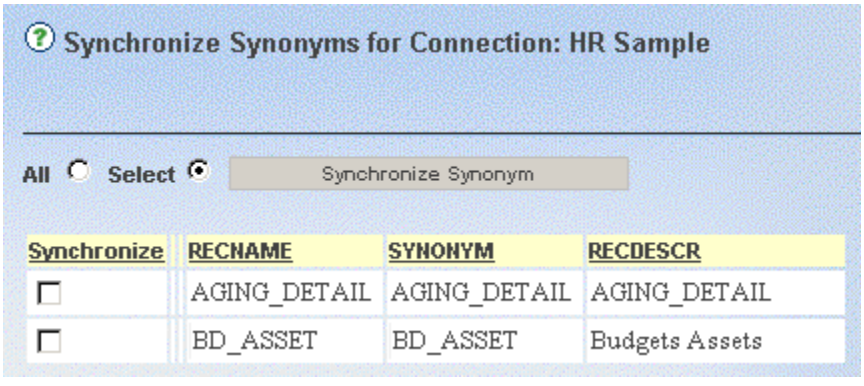
- Queries the internal repository to determine which records were installed for the selected PeopleSoft connection.
- Compares the version number in those records with the version number in the PeopleSoft data source. If none of the records have changed, a message appears.

If there are any records that have changed, they appear in the Select Record(s) to Synchronize page.

Procedure: How to Re-synchronize Synonyms

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Re-Synch* under the Synonyms group.

   The Synchronize Synonyms for Connection window opens.



3. Select the check box in the Synchronize column for each synonym you want to re-synchronize with PeopleSoft Record Definitions, or select *All* to check all listed synonyms.
4. Click *Synchronize Synonym*.

Refreshing Synonyms

After creating your synonyms, you may need to fresh them at some point.

Procedure: How to Refresh a Synonym

1. From the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down menu. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Refresh* under the *Synonyms* group.

   The Refresh Synonyms for Connection window opens.
3. Select the synonyms you want to fresh under the *Refresh* column, or select the *All* option.
4. Deselect the *Resynch Security File* check box if you have more synonym administration to do during this session. Leave it checked for the last synonym administration task.
5. Click *Fresh Synonym*.

Viewing Sample Data

After creating a synonym, you may want to verify that it is using the correct Record Definition. The *Synonyms*, *View* option on the PeopleSoft menu provides an easy method for listing the existing synonyms and viewing a sample data set from each.

Renaming a Synonym

You can selectively rename a synonym using the *Synonyms*, *Edit* option on the PeopleSoft menu.

After renaming a synonym, you must update references in existing reports from the old name to the new name using appropriate options in your reporting tool.

Managing PeopleSoft Secured Data Access

An administrator can manage security access according to individual users or a permission list. The decision for how to manage access is made as a connection option, and the two methods are mutually exclusive.

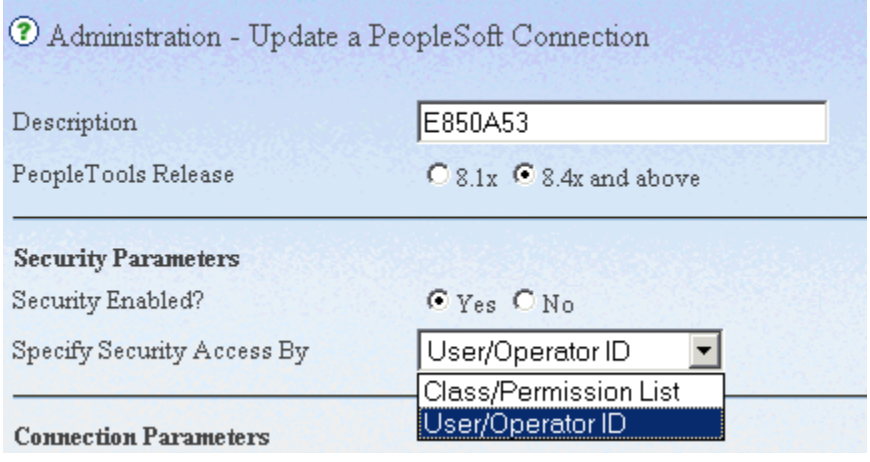
Regardless of the method for granting access rights to users, the process for managing security access is roughly identical. The administrator can affect security access for a user in the following ways:

- Adding security access.
- Removing security access.
- Resynchronizing security access.

Enabling Access

Before a PeopleSoft user can run reports against PeopleSoft, security access rights must be granted. The PeopleSoft connection does not automatically enable reporting for every user; instead an administrator must decide which users or permission lists have reporting access.

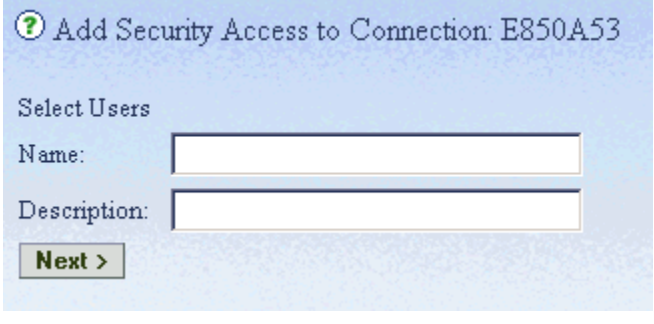
Note: Switching between users and permission lists is controlled by the *Specify Security Access By* drop-down menu under *Security Parameters* on the Update a PeopleSoft Connection window.



Procedure: How to Enable Security Access

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Add Access* under the Security group.

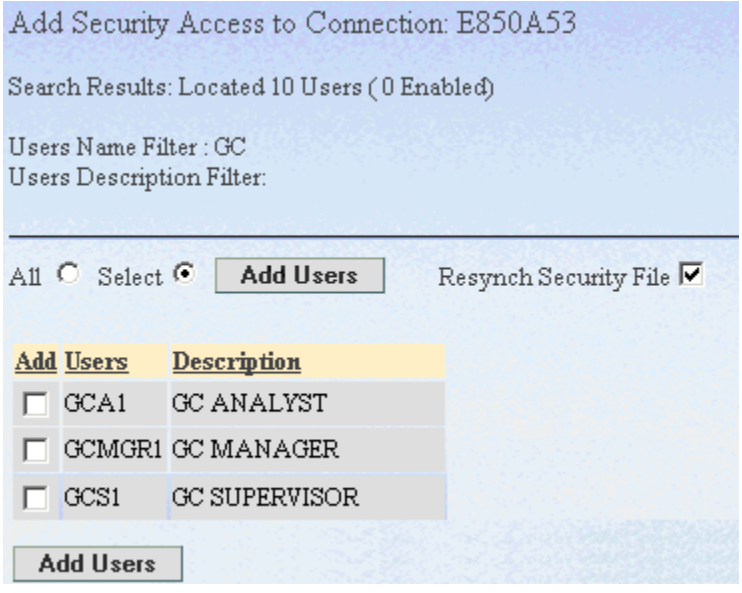
   The Add Security Access to Connection window opens.



   Note: If the security parameter of your connection is set to *Class/Permission Lists*, the window will allow you to *Select Permission Lists* rather than *Select Users*.
3. Select filtering criteria to reduce the list of users or permission lists, and click *Next*, or leave the fields blank and click *Next* to see a full list.

   Note: If you do not use a filter, or your filtered results are very large, the results may be truncated. Use a different filter to reduce the returned results.

The Add Security Access to Connection window opens.



4. Select the check boxes in the *Add* column for the users or permission lists you would like to add access for, or select *All* to check all those listed.
5. Click *Add Users* or *Add Permission Lists* to add your selections.

Note: When you add users by Permission List, the users will only be granted access to Records with which the added Permission Lists are associated.

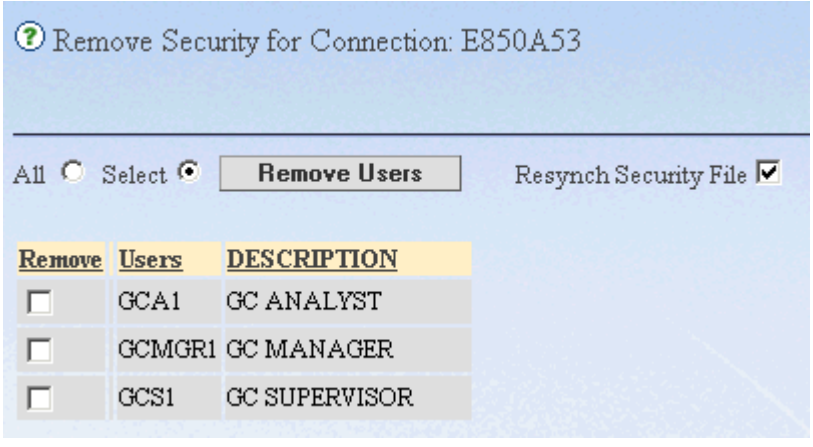
Disabling Access

You can remove security access from the PeopleSoft menu.

Procedure: How to Remove Security Access

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Remove Access* under the Security group.

The Remove Security for Connection window opens.



Note: If the security parameter of your connection is set to *Class/Permission Lists*, the window will allow you to *Select Permission Lists* rather than *Select Users*.

3. Select the check boxes in the Remove column for the users or permission lists that should be removed, or select *All* to check all those listed
4. Click *Remove Users* or *Remove Permission Lists*.

   The PeopleSoft connection removes all references to these users or permission lists. After they are removed, the associated users cannot access PeopleSoft data through the PeopleSoft adapter.

Updating Access

You can re-synchronize PeopleSoft security metadata when changes occur in the PeopleSoft environment. The following situations require resynchronization:

- A permission list has had query access group privileges modified.
- A Record has been added or removed from a query tree.
- A user has been added to or removed from a permission list with security access enabled.
- Row-level security class or permission lists are changed.

You do not need to run the security resynchronization routine if row-level security access has changed, based on the security table being updated. This type of security change will be enforced automatically and immediately. If the Query Security Record within a Record Definition has been changed, the synonym must be updated.

Note: A batch mechanism for resynchronization is also available that may be scheduled through a command line execution. For more information, see *Advanced Administrative Topics* on page 1886.

Procedure: How to Re-synchronize Security Rules

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Re-Synch* under the Security group. Security rules are re-synchronized based upon security access settings.

   Processing may take several minutes. Wait for a completion message before attempting other administrative tasks.

Managing Connections to PeopleSoft

Multiple connections to PeopleSoft data sources are possible from within one server configuration. However, depending on your reporting requirements, it may be optimal to create separate configurations for each data source.

Creating multiple connections using the same server configuration is a good option under the following circumstances:

- The site has two or more PeopleSoft application suites and must perform significant combined reporting. A typical example is a site that has both PeopleSoft HRMS and PeopleSoft Financials applications where you must display data from both systems in a single report.
- Two or more PeopleSoft data sources that must be accessed reside on the same system. If the data sources must be accessed for display in a single report output and if the server is located on the same computer system, this architecture can provide optimal performance.
- Two or more PeopleSoft data sources reside in different locations with requirements for moderate to significant consolidated reporting. This architecture provides optimal effectiveness in report development, system maintenance, and performance.

Create separate configurations instead of using multiple connections in these situations:

- Only a single PeopleSoft production data source must be supported. Each development and test instance should have a separate configuration.
- Multiple PeopleSoft data sources exist with only a limited requirement for combining data in reports.

The remaining instructions in this topic assume a single configuration directory structure for one or more data source connections.

Updating a PeopleSoft Connection

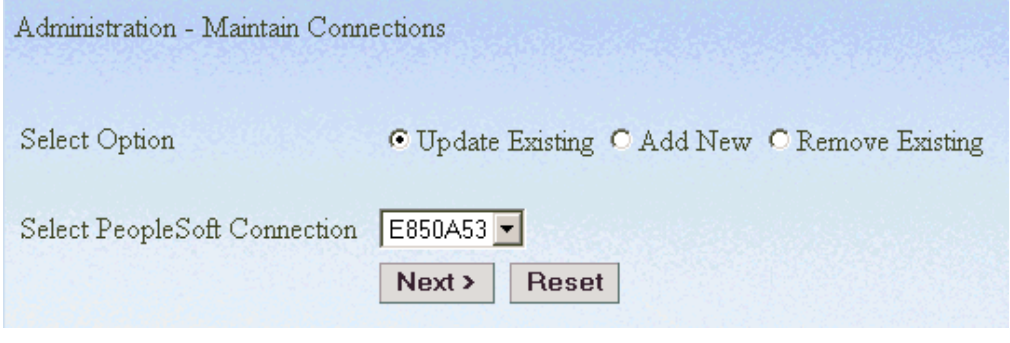
If it becomes necessary to update the connection parameters to a PeopleSoft data source, you can use Administration options to perform the following functions:

- Turn PeopleSoft data access security enforcement on or off.
- Toggle between trusted and password authentication.
- Manage the user access specification type.
- Manage all data source identification information.
- Define default synonym behavior.

Procedure: How to Edit a PeopleSoft Connection

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Connections* under the *Administration* group.

   The Administration - Maintain Connections window opens.



3. Select *Update Existing*.
4. Select a connection from the *Select PeopleSoft Connection* drop-down list.
5. Click *Next*.

   The Administration - Update a PeopleSoft Connection window opens. For details, see *Administration - Create a New PeopleSoft Connection Window* on page 1857.

6. Make the necessary modifications, test the connection, and if successful, click *Configure*. If not successful, edit the information until the test is successful, then click *Configure*. For more information, see step 5 in *How to Add a PeopleSoft Data Source Instance* on page 1881.

   Note: The *Configure* button becomes available once the connection test to the selected RDBMS is successful.

7. On the next screen, click *Next*.

Reference: Updating Connections

When you change the *Specify Security Access By* setting, you must re-synchronize security before changes will take effect. We suggest that you remove all currently configured users before you make this change, and re-add them after security settings have been modified.

As when you modify authentication information, you must be careful to manage changes to data source connectivity correctly. Always modify your settings in the PeopleSoft Connection first, then change them at the data source. Where possible, keep both access options in place during a configuration conversion process until tests have verified a successful switch.

Adding a PeopleSoft Connection

After you add your first connection, you can specify additional PeopleSoft connections. Note that during this process, you are not prompted for the DBA information prior to managing metadata and connection information since all additional PeopleSoft data sources must use the same DBA in order for base metadata to function properly.

Procedure: How to Add a PeopleSoft Data Source Instance

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Connections* under the *Administration* group.

   The Administration - Maintain Connections window opens.
3. Select *Add New*.

   Note: If there are currently no connections, *Add* will be the only option.
4. Click *Next*.

   The Administration - Create a New PeopleSoft Connection window opens. For details, see *Administration - Create a New PeopleSoft Connection Window* on page 1857.

5. Type the necessary information and click *Test Connection*. If

   - The connection is successful, the following message will appear in green to the right of the button:

     ```
     Connection successful
     ```

   - The connection is unsuccessful, the following message will appear in red to the right of the button with the relevant RDBMS return code:

     ```
     Connection failed - return code: nnnn
     ```

   When the test is successful, the *Configure* button appears.

6. Click *Configure*.
7. On the next screen, click *Next*.

When processing finishes, you can manage your synonyms and security for this connection.

Removing a PeopleSoft Connection

The integration with PeopleSoft depends on valid connection information. If a previously created connection is no longer required, you should remove that connection.

Procedure: How to Remove a PeopleSoft Connection

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Connections* under the *Administration* group.

   The Administration - Maintain Connections window opens.
3. Select a connection from the *Select Connection* drop-down list.
4. Select *Remove Existing*.
5. Click *Next*.

   The Administration - Remove a PeopleSoft Connection window opens.

   You are prompted to confirm. After confirmation, the system removes the selected PeopleSoft Connection and all of its associated metadata.

Using Administrative Utilities

The Adapter for PeopleSoft provides administrative tools that enable you to report on connection information, update the DBA Password, and set adapter tracing options.

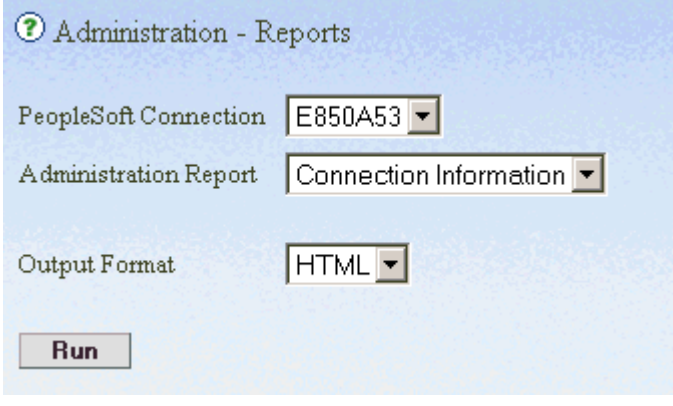
Connection Reports

The Administrator provides reports to make administration easier. You can run reports that provide information on connections, metadata, users, and user access. These reports are accessible from the index.

Procedure: How to Run Connection Reports

1. In the PeopleSoft menu, select the connection you want to manage from the *Select Connection* drop-down list. For details on accessing the PeopleSoft menu, see *Accessing the Adapter for PeopleSoft Administrator* on page 1865.
2. Click *Reports* under the Administration group.

   The Administration - Reports window opens.



3. If necessary, select the PeopleSoft connection you want to run a report for from the *PeopleSoft Connection* drop-down list.
4. Select a report type from the Administration Report drop-down list. The options are:
   - **Connection Information.** Retrieves statistics for your connection.
   - **Synonyms.** Retrieves a list of available synonyms.
   - **Users/Oprids.** Retrieves a list of users who have access to the current connection.
   - **Access Assigned.** Retrieves a list of the access assigned to each user.
5. Select an output format for the report from the Output Format drop-down list. The options are HTML and PDF.
6. Click *Run*.

The selected report appears.

Updating the DBA Password

The DBA Password option is used for updating the password that is supplied during the initial configuration of the adapter. When the password is updated, all synonyms are updated as well. (Note that this process may take several minutes.)

Tracing Adapter Processing

The adapter tracing utility creates application level traces for support purposes. Typically, tracing is used under the direction of Information Builders Customer Support Services.

Log off

This option, when clicked, will provide a confirmation dialog box to log off from the adapter and free the server agent. Select *OK* or *Cancel*.

Note: Closing the window will have the same effect (without the confirmation box).

Migrating from 7.1x and 7.6.x to 7.7

After installing the 7.7 server, follow these steps for migrating the PeopleSoft Adapter environment from 7.1.x or 7.6.x releases.

1. Create the following three directories under the 7.7 server's approot directory.
   - ❑ snapcat
   - ❑ snapcat/bak
   - ❑ snapinst
2. Copy the contents of the old snapinst and snapcat folders to the new directories.
3. Update the Application Path to include snapcat.
4. Configure the relevant SQL Database Adapter.
5. Configure the PeopleSoft Adapter.
6. Proceed to *Adding the First Connection to PeopleSoft* on the DBA screen, and enter the DBA password that you used in the earlier release.

The PeopleSoft adapter is now ready for use.

Note: If you are migrating from a server release earlier than 7.7.04 to release 7.7.04 (or later), the PSLOGIN procedure will be added to each synonym, as an MFD_PROFILE entry that is registered to the PeopleSoft adapter. This process will take place every time the adapter is used using the Web Console. Existing synonyms will be backed up in the SNAPBAK application directory, and a log file will be created in the same directory that contains migration statistics. The log file name takes the format *psmiglog_hh.mm.ss* and will be created every time that the adapter is used. It is recommended that older log files are deleted periodically.

In WebFOCUS Release 82 (Reporting Server Release 7.7.07) and higher, the automatic migration also adds a global variable to each PeopleSoft synonym. This variable is coded on the CONNECTION= parameter in the ACCESS (synonymname.acx) file. This is used to facilitate the moving of the PeopleSoft adapter files from one server to another, for example, Test to Production, where the connections (instances of the PeopleSoft environment) may have different names. For more information, see *Moving the PeopleSoft Adapter from Server to Server* on page 1885.

Moving the PeopleSoft Adapter from Server to Server

First, follow the steps outlined in *Migrating from 7.1x and 7.6.x to 7.7* on page 1884. If the connection names are the same, the adapter is ready to be used. If the connection names are different, you must also perform the following steps:

1. Select *Connections* in the Administration section of the adapter.
2. Select *Update Existing* for each configured connection.
3. Change the Database Identifier (and Server, if MS SQL Server is used) to point to the correct PeopleSoft instance for the connection. If required, make changes to any other parameters at this time.
4. Click the *Test Connection* button.
5. If the test is successful, click *Configure*.

Once these steps are completed for all configured connections, the PeopleSoft adapter is ready for use.

Changing the WebFOCUS Reporting Server Code Page to Unicode

Note: It is highly recommended that you contact the Information Builders Customer Support Services before changing the server code page. They will be able to help or explain the steps required to move the PeopleSoft adapter to the new code page environment.

The PeopleSoft adapter uses a number of FOCUS database files to store adapter information, which will no longer be readable using the new code page. Prior to the change, the following steps should be taken:

1. Backup the *snapcat* and *snapinst* directories.
2. Save the connection parameters for the adapter by capturing the information shown in the Connection window. Click *Connections* under the Administration group.
3. Select *Update Existing* and then select a connection from the Select PeopleSoft Connection drop-down menu.
4. Click *Next*. Take a screen shot of the connection parameters for each connection. There is no need to actually update the connection.
5. Make a note of all users for each connection.
6. Make a note of all PeopleSoft tables for each connection.
7. Delete all *.mas* and *.acx* files in snapcat directory.
8. Delete the *snapinst* directory.
9. Change the code page. Doing this will restart the server.
10. Configure the PeopleSoft adapter for initial connection.
11. Configure all other connections.

Advanced Administrative Topics

In addition to the standard administrative options described in this chapter, advanced options and techniques are available to assist in automating security access updates and troubleshooting integration problems. This topic discusses these options and provides additional details on PeopleSoft security integration.

Automating Security Access Updates

Security resynchronization is one of the functions of the PeopleSoft Administrator. When administering PeopleSoft connections through the Web console, you simply select the Re-Synch link under the Security group, and all of the PeopleSoft data security rules are re-synchronized.

The batch resynchronization utility enables administrators to schedule, through an operating system command, the execution of this functionality without having to manually open the Administrator tool. By scheduling this process nightly (or otherwise; based on business requirements), administrators are not required to perform this function manually, and they can be assured that the data security rules are current as of the last scheduled run.

The utility is comprised of two files. These files are a procedure (pssecsnk.fex) used in executing the resynchronization routine and a t3i script (pssecsnk.t3i), which is the command line input file. By starting the server with the t3i script as an input, the server performs the required security resynchronization without user intervention.

A server can be accessed locally without the use of client software. This functionality can be started using the following mode:

- **Batch Mode** is useful for scheduling batch scripts with operating system tools.

An administrative user with access to the server can use the server in stand-alone mode to perform any of these functions. If PeopleSoft data must be accessed in any of these processes, you must execute the PeopleSoft authentication routine.

Procedure: How to Run Batch Mode Security Resynchronization From the Command Line

The following procedure is for a Windows environment.

1. Create a temporary work directory outside of the IBI directory structure, and then navigate to it.
2. Execute the security synchronization script command or create a .bat file that calls it (for example, pssecsynch.bat, which would use the “call” keyword in front of the following command). The synchronization script command syntax is as follows:

   ```
   c:\ibi\srv77\wfs\bin\edastart -f c:\ibi\srv77\wfs\catalog\pssecsnk.t3i
   ```

   where:

   `c:\ibi\srv77\wfs\bin\edastart`

   Is the directory location of the edastart command.

For details about securing the script, see *Password Security* on page 1888.

Note: The pssecsnk.t3i script will create a number of work files in the temporary directory. Optionally, these can be deleted using the following four commands, which are placed in the .bat file after the call to the t3i script:

- del *.mas
- del *.ftm
- del *.fex
- del *.foc

Reference: Password Security

The t3i script is an ASCII text file that must be edited for your particular environment. It contains four execute command lines that:

- Load the PeopleSoft adapter system metadata into memory.
- Log into PeopleSoft.
- Perform the security resynchronization.
- Perform a batch Logoff from the adapter.

The syntax is:

```
%connect
%begin
APP PREPENDPATH SNAPINST
EX _EDAHOME/CATALOG/PS/PSMASLOD
EX _EDAHOME/CATALOG/PS/PSLOGOFF RETVAL=N, PSBATCH=Y
EX PSLOGIN USERID=PS, PASSWD=PS, PSBATCH=X
EX _EDAHOME/CATALOG/PS/PSSECBAT
EX _EDAHOME/CATALOG/PS/PSLOGOFF RETVAL=N
%end
%disconnect
%stop_server
```

You must modify the user ID and password in accordance with one that is appropriate for your environment. The user ID must be granted access in PeopleSoft.

Since the file is in readable text, the administrator must prevent read/write access for anyone not authorized to access the t3i. The best way to prevent unauthorized access is to use operating system security. Windows and UNIX provide ways to prevent unauthorized access to individual files and entire directories. Contact your operating system administrator for assistance in performing this security step.

Procedure: How to Capture Execution Results

After the script has been executed, the Administrator can review the execution results. A batch process output file (pssecsnk.t3o) is automatically created in the execution directory.

Example: Capturing Execution Results

The following is an example of a successful execution. Look at the bottom of the following file:

```
< Filename: pssecsnk.t3o >
…
Various REBUILD message lines
…
Successful Security Re-synch for Connection: E854064
DBNUM:  1 (E854064) has been re-synchronized, DBA security is: ON
```

Using Cluster Synonyms

The PeopleSoft adapter administration screens allow for the creation of base synonyms for individual PeopleSoft records. Each created synonym will use an MFD_PROFILE keyword to execute a procedure to dynamically create a security file and a DBA section that references the security file. PeopleSoft synonyms should not be changed Refreshing them using the administration screens will remove the changes.

A cluster synonym can be created that references the base synonym . Changes or additions can be made to this cluster. The cluster can also JOIN PeopleSoft synonyms together. If clusters are used, the following rules must be applied:

- Do not include the keyword MFD_PROFILE=PSLOGIN
- Do not include a DBA entry.
- Always use the *Reference to existing synonym* option when creating the cluster.

Troubleshooting Tips

These troubleshooting tips help if you are getting FOCUS error messages, such as a FOC1302, or if you are seeing SQL errors when you run reports.

Procedure: How to Connect to the PeopleSoft Data Sources

The most likely source of difficulty in connecting to your PeopleSoft data source is the data source connectivity. If you are experiencing difficulty connecting to the data source:

1. Verify that the data source instance is up and running.
2. Run a client query tool on the server, using the PeopleSoft access ID.
   - If you cannot log in to the data source, there is a problem with the access ID.

     Verify the ID and check to see if the password has changed.

     If the password has changed, then it must also be changed in the PeopleSoft configuration.
   - If you can log in, proceed to Step 3.

3. Run a few simple queries against the PeopleSoft records that are returning errors.

   If you are not getting any rows back, check to see if the PeopleSoft access ID has read authority on the PeopleSoft records you are accessing.

Procedure: How to Verify Security

PeopleSoft enforces all the security mechanisms used by the PeopleSoft Query tool. This includes Query Tree security and row-level security. If you are getting unexpected results, it is possible that your PeopleSoft security has not been configured properly.

To verify that security is being properly enforced:

1. Verify that the synonym(s) being queried are correct.

   The Administrator can run a report of installed records to verify. If row-level security is the issue, the server-side ACX file should be reviewed and compared to the record definition in PeopleSoft, including the Security Search record.

2. Create a similar query in PS/Query. The SQL code can be viewed and compared to SQL tracing on the server.
3. Examine the SQL.

   Does it appear to match the business logic behind the original request? If not, check the syntax behind the original FOCUS code. Perhaps you are using a WRITE instead of a PRINT, or perhaps a WHERE clause has been coded incorrectly.

4. Run this SQL directly against the RDBMS using a data source query tool.

   If you are getting the expected results, then there may be a problem with the program logic after the data is returned to FOCUS from the RDBMS. Examine COMPUTE/DEFINE statements for logic errors or, if you are using ON TABLE HOLD, there may be a problem in your code.

   If you are getting the same unexpected results, then examine the underlying data. Perhaps some cross-referenced fields are missing, or perhaps row-level security is eliminating essential rows.

Chapter 78

Using the Adapter for PostgreSQL

The Adapter for PostgreSQL allows applications to access PostgreSQL data sources. The adapter converts application requests into PostgreSQL calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the PostgreSQL Environment

In order to use the Adapter for PostgreSQL, you must install the PostgreSQL driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the PostgreSQL Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server EDAPRINT log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for PostgreSQL

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a PostgreSQL data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf), if supported on your platform.

You can declare connections to more than one PostgreSQL data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for PostgreSQL

The *PostgreSQL* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the PostgreSQL data source.

Driver name

Name for the PostgreSQL JDBC driver.

For example: org.postgresql.Driver

See PostgreSQL documentation for the specific driver release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLPSTGR SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

SQLPSTGR

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name.

URL

Is the URL to the location of the PostgreSQL data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the PostgreSQL Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMS will encrypt the password before adding it to the server profile.

Note: Consult vendor documentation for the exact name, port, and path.

```
ENGINE SQLPSTGR SET CONNECTION_ATTRIBUTES CON1
  'jdbc:postgresql://hostname:5432/qatst'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLPSTGR SET DEFAULT_CONNECTION connection
```

where:

SQLPSTGR

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLPSTGR SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLPSTGR SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLPSTGR`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing PostgreSQL Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLPSTGR to identify the Adapter for PostgreSQL.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLPSTGR [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLPSTGR`

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using PostgreSQL, you must locally install the RDBMS PostgreSQL Driver.

The Server can access third-party database tables across the PostgreSQL network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Note: If you are creating a synonym for a PostgreSQL data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for PostgreSQL

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *PostgreSQL Data Type Support* on page 1904.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for PostgreSQL synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLPSTGR ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the PostgreSQL table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner*.]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

PostgreSQL Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Data Type Mapping for A256V

The Adapter for PostgreSQL supports data type mapping using the following setting:

```
ENGINE SQLPSTGR SET CONVERSION LONGCHAR ALPHA n
```

The setting affects the mapping of all PostgreSQL native fields that have a data type of [VAR]CHAR(32767) or TEXT. The value of *n* ranges from 1 to 32767. The default value is 256. The TEXT data type (no *n* length) can be set instead of ALPHA when necessary.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the PostgreSQL Environment

The Adapter for PostgreSQL provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to PostgreSQL.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLPSTGR SET TIMEOUT {nn|0}
```

where:

`SQLPSTGR`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLPSTGR SET PASSRECS {ON|OFF}
```

where:

`SQLPSTGR`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

PostgreSQL Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 79

Using the Adapter for Presto

The Adapter for Presto is a distributed SQL query engine designed to query large data sets. The adapter converts application requests into JDBC calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Presto Environment

In order to use the Adapter for Presto, you must install the JDBC driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the Presto JDBC Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, issue the following commands:

   ```
   CLASSPATH=/usr/presto-jdbc-0.218.jar;
   $CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\presto-jdbc-0.218.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

(Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and checking that the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server EDAPRINT log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for Presto

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Presto

The *Presto* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Presto data source in the following format:

```
jdbc:xxxxxxx://hostname:port/datasource
```

For SSL, use the following format:

```
jdbc:xxxxxxx://hostname:port/datasource?SSL=true&
   |SSLKeyStorePath=/path/cert/presto.jks&
   |SSLKeyStorePassword=sslpassphrase
```

Driver name

Name for the Presto driver.

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.

- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLPRS SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLPRS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the Presto data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the Presto Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLPRS SET CONNECTION_ATTRIBUTES CON1
  'jdbc:xxxxxxx://hostname:port/datasource'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLPRS SET DEFAULT_CONNECTION connection
```

where:

SQLPRS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLPRS SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLPRS SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLPRS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Presto Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Presto

The following list describes the synonym creation parameters for which you can supply values.

Object Type

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Owner/Schema

Selecting this option adds the Owner/Schema and Object Name parameters to the screen. Select an owner/schema from the drop-down list or type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. Then click *Search*. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

Object Name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen. Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. Then click *Search*. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Object Type* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Row Limit

Select the number of objects to display on the Create Synonym page.

For Subquery

Only available when *External SQL Scripts* is selected from the object type drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 1211.

Create: Cluster Synonym or Base Synonyms

Select the button for the type of synonym you want to create.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Fact or Dimension

You can check Fact or Dimension to generate the segment as a fact table or a dimension segment. If you are creating a cluster synonym, you can right-click a selected fact table and select *Show Related Dimensions* or *Add Related Dimensions* to show a list of related dimensions or add related dimensions to the synonym.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- When creating base synonyms, you can select all tables in the list by clicking the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Once you have selected the objects for which you want to create synonyms, click the *Create Base Synonyms* or *Create Cluster Synonym* button on the ribbon.

Example: Sample Generated Synonym

An Adapter for Presto synonym comprises a Master File and an Access File. This is a synonym for the table nf29104.

Master File nf29104.mas

```
FILENAME=NF29104, SUFFIX=SQLPRS  , $
  SEGMENT=NF29104, SEGTYPE=S0, $
    FIELDNAME=DIVISION4, ALIAS=division4, USAGE=I11, ACTUAL=I4,
      MISSING=ON, $
    FIELDNAME=DIVISION_NA4, ALIAS=division_na4, USAGE=A25V, ACTUAL=A25V,
      MISSING=ON, $
    FIELDNAME=DIVISION_HE4, ALIAS=division_he4, USAGE=I11, ACTUAL=I4,
      MISSING=ON, $
```

Access File nf29104.acx

```
SEGNAME=NF29104,
   TABLENAME="r729999d"."nf29104",
   CONNECTION=CON1, $
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Presto table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support Report

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Presto Environment

The Adapter for Presto provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Presto.

Syntax: **How to Issue the TIMEOUT Command**

```
ENGINE SQLPRS SET TIMEOUT {nn|0}
```

where:

`SQLPRS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native Presto driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLPRS SET PASSRECS {ON|OFF}
```

where:

`SQLPRS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Presto Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 80

Using the Adapter for Progress

The Adapter for Progress allows applications to access Progress data sources. The adapter converts application requests into native Progress statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Progress Environment

Currently, the Progress environment is supported on UNIX and Windows. The Adapter for Progress is available as an ODBC interface.

Procedure: How to Set Up the Environment on Windows Using ODBC

On Windows, the Progress environment is set up during the installation of the product.

Procedure: How to Set Up the Environment on UNIX Using ODBC

1. Specify the full name of the directory containing Progress.

   ```
   DLC=/rdbms/pro91d/dlc
   ```

2. Specify the full name of the default Progress startup file.

   ```
   PROSTARTUP=/rdbms/pro91d/dlc/startupyy.pf
   ```

3. Specify the location of the ODBC initialization file.

   ```
   ODBCINI=/usr/odbc/ibiodbc/progress.ini
   ```

4. Specify the path to the Progress shared libraries.

   ```
   LIBPATH=/rdbms/pro91d/dlc/odbc/lib:/rdbms/pro91d/dlc/lib
   ```

Configuring the Adapter for Progress

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Connecting to a Progress Database Server

For an ODBC interface, the server must be brought up in TCP/IP mode.

Declaring Connection Attributes

In order to connect to a Progress database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Progress database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Progress Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Progress Using ODBC

The *Progress* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Authorization ID to connect to the source.

Password

Password associated with the authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLPRO SET CONNECTION_ATTRIBUTES connection
DSN_name/userid,password
```

where:

`SQLPRO`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the Progress Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`userid`

Is the primary authorization ID by which you are known to Progress.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the Progress DSN named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLPRO SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior to Version 7 Release 6.1, the connection was defined as:

  ```
  ENGINE SQLPRO SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

```
ENGINE SQLPRO SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLPRO SET DEFAULT_CONNECTION connection
```

where:

`SQLPRO`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLPRO SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: **How to Control the Connection Scope**

```
ENGINE SQLPRO SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLPRO`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Progress Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Progress data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Progress table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Progress

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Progress Data Type Support* on page 1936.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Progress synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLPRO ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4 ,TABLENAME=EDAQA.NF29004,
KEYS=1, WRITE=YES ,$
```

Reference: Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: `TABLENAME=[location.][owner.]tablename`

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Keyword	Description
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Progress Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Controlling the Mapping of Variable-Length Data Types for SQL-92

The SET parameter VARCHAR controls the mapping of the Progress data type VARCHAR. By default, the server maps these data types as variable character (A*n*V).

The following table lists data type mappings based on the value of VARCHAR:

Progress Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE SQLPRO SET VARCHAR {ON|OFF}
```

where:

SQLPRO

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Progress data type VARCHAR as variable-length alphanumeric (A*n*V). ON is the default value.

OFF

Maps the Progress data type VARCHAR as alphanumeric (A).

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLPRO SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Manipulating Arrays Created in the 4GL Environment Under SQL-92

When the Adapter for Progress creates a synonym from an array in the Progress 4GL environment, the array values are translated into a semi-colon delimited list. This is dictated by limitations on array support by the Progress client API for both metadata and processing.

Example: Sample Master File With Arrays

In this example the ARRAY column is called ITEMS. The metadata generated by the Adapter for Progress (under SQL-92/ODBC) represents ITEMS using the data type VARCHAR. The Adapter for Progress queries the Progress field SQL_WIDTH (which sets field widths to twice the length of the displayed format) in order to determine the USAGE and ACTUAL fields lengths in the synonym.

One additional step is required to retrieve data from the values in the synonym: you must manipulate the array using the WebFOCUS function GETTOK, which divides a character string into substrings (called tokens). A specific character, called a delimiter, occurs in the string and separates it into tokens. GETTOK returns the token specified by the token_number. For example, to extract the second value from five defined in an array, you would specify 2 as the token_number and the semi-colon character as the delimiter. GETTOK would then divide the string into substrings using this delimiter and extract the second value.

Following is a synonym containing the array field ITEMS followed by a two procedures: the first produces a semi-colon delimited list; the second uses the GETTOK function to retrieves data from the array field.

Synonym

```
FILENAME=ARRAYTAB, SUFFIX=SQLPRO   , $
  SEGMENT=ARRAYTAB, SEGTYPE=S0, $
   FIELDNAME=CUSTOMER, ALIAS=customer, USAGE=A40V, ACTUAL=A40V,
      MISSING=ON, $
    FIELDNAME=ITEMS, ALIAS=items, USAGE=A60V, ACTUAL=A60V,
      MISSING=ON, $
```

Basic Procedure

Since the concept of an array, with five distinct items (item1=val1; item2=val2; etc.) does not exist for Progress in the SQL-92 environment, the item values for customer 1 are represented as a semi-colon delimited list.

```
TABLE FILE ARRAYTAB
PRINT *
END
```

The output is:

```
CUSTOMER                        ITEMS
--------                        -----
customer1                       val1;val2;val3;val4;val5
```

Procedure With GETTOK

In this procedure, the GETTOK function is used to retrieve a single value from the array

```
TABLE FILE ARRAYTAB
PRINT CUSTOMER AND COMPUTE
ARRAY/A5 = GETTOK(ITEMS, 60, 2, ';', 5, ARRAY);
END
```

where the function arguments are:

GETTOK argument value	Is...
ITEMS	A field in the synonym.
60	The length of the ITEMS field.
2	The number of the token (value) to extract from the array.
;	The delimiter character enclosed in single quotation marks (which must be a semi-colon in this context).
5	The maximum length of the field (in this example, it must match the length of ARRAY/A5).
ARRAY	The field in which to store the generated output.

The output is:

```
CUSTOMER                        ARRAY
--------                        -----
customer1                       val2
```

For details about the GETTOK function, see the *WebFOCUS Using Functions* manual.

Customizing the Progress Environment

The Adapter for Progress provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode

The Adapter for Progress has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE SQLPRO SET NONBLOCK {0|n}
```

where:

`SQLPRO`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLPRO SET PASSRECS {ON|OFF}
```

where:

SQLPRO

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Specifying a Time-out Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Progress.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLPRO SET TIMEOUT {nn|0}
```

where:

SQLPRO

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

0

Represents an infinite period to wait for a response.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the SET ISOLATION command.

Syntax: How to Specify Transaction Isolation Level From a SET Command

You can specify transaction isolation level by issuing the following command

```
ENGINE SQLPRO SET ISOLATION {RU|RC|RR|SE}
```

where:

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

Progress Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Specifying Block Size for Retrieval Processing

The Adapter for Progress supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE SQLPRO SET INSERTSIZE n
```

where:

SQLPRO

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Chapter 81

Using the Adapter for PSQL

The Adapter for PSQL allows applications to access PSQL data sources. The adapter converts application requests into PSQL calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the PSQL Environment

In order to use the Adapter for PSQL, you must install the PSQL driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the PSQL Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server's EDAPRINT. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for PSQL

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a PSQL data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one PSQL data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for PSQL

The PSQL adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the PSQL data source.

Driver name

Name for the PSQL JDBC driver.

See PSQL documentation for the specific driver release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLPSQ SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

SQLPSQ

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name.

URL

Is the URL to the location of the PSQL data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a books data source using the PSQL Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

Note: Consult your vendor documentation for the exact name, port, and path.

```
ENGINE SQLPSQ SET CONNECTION_ATTRIBUTES CON1
  'jdbc:pervasive://edared30:1583/DEMODATA
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLPSQ SET DEFAULT_CONNECTION connection
```

where:

SQLPSQ

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLPSQ SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLPSQ SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLPSQ`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing PSQL Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLPSQ to identify the Adapter for PSQL.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLPSQ [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLPSQ`

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using PSQL, you must locally install the RDBMS PSQL Driver.

The Server can access third-party database tables across the PSQL network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note: If you are creating a synonym for a PSQL data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for PSQL

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 1958.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for PSQL synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLPSQ ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.

Keyword	Description
TABLENAME	Identifies the PSQL table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the PSQL Environment

The Adapter for PSQL provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to PSQL.

Syntax: **How to Issue the TIMEOUT Command**

```
ENGINE SQLPSQ SET TIMEOUT {nn|0}
```

where:

`SQLPSQ`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: **How to Cancel Long Requests**

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLPSQ SET PASSRECS {ON|OFF}
```

where:

`SQLPSQ`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

PSQL Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 82

Using the Adapter for Python

Python is a high-level, easy to use, powerful, interpreted programming language suitable for scripting, as well as complex programming.

The Python standard library, an extensive collection of modules, implements the Python "batteries included" philosophy that gives programmers immediate access to sophisticated and robust capabilities that make it easy to write your own Python functions to be used in WebFOCUS.

The Adapter for Python defines a connection to the Python interpreter for executing user-written Python scripts that generate calculated fields (WebFOCUS Computes). These fields can be used in WebFOCUS Workbooks, WebFOCUS InfoGraphics, charts, reports, dashboards, and WebFOCUS portals.

In this chapter:

Guidelines for Writing Python Scripts to be Used With WebFOCUS

The following Python script, named arithmetic_example.py contains a function named adder that adds two numbers and conforms to the requirements described in this section.

```
# arithmetic_example.py

import csv


def adder(csvin, csvout):

    with open(csvin,  'r', newline='') as file_in,\
         open(csvout, 'w', newline='') as file_out:

        fieldnames = ['addition']

        reader = csv.DictReader(file_in, quoting=csv.QUOTE_NONNUMERIC)

        writer = csv.DictWriter(file_out, quoting=csv.QUOTE_NONNUMERIC,
                                fieldnames=fieldnames)
        writer.writeheader()

        for row in reader:
            addition = row['a_number'] + row['another_number']

            writer.writerow({'addition': addition})
```

A Python script must conform to the following requirements in order to be compatible with WebFOCUS.

- **csv module requirement.** The script must import the csv module because WebFOCUS sends data to the Python script using an automatically generated, temporary .csv file. The name of the file is stored in a global Python variable named *csvin*. The global Python variable *csvout* contains the name of the temporary .csv results file to be returned to WebFOCUS. WebFOCUS sets the values for csvin and csvout, and they should not be changed by the programmer. The temporary files are removed immediately and cannot be viewed by the user. The format of both files is the same. The default delimiter is a comma (,) and cannot be changed. All non-numeric fields are enclosed in double quotation marks.

  The global variables *csvin* and *csvout* are defined by WebFOCUS for reading and writing .csv files.

- **csvin and csvout format parameter.** The script must open csvin and csvout with the following format parameter:

  ```
  newline=''
  ```

If newline='' is not specified, newlines embedded inside quoted fields will not be interpreted correctly, and on platforms that use \r\n line endings on write, an extra \r will be added.

- **reader and writer format parameter.** The following format parameter should be included in the reader and writer statements, whether you are using reader and writer or DictReader and DictWriter, to enclose non-numeric values in double quotation marks and make it clear which values are numeric (they will be converted to floating point values):

  ```
  quoting=csv.QUOTE_NONNUMERIC
  ```

  This indicates that non-numeric (alphanumeric, text, string, and date) values are enclosed in double quotation marks and that numeric values are not. When *csvin* is read, all WebFOCUS numeric values will be automatically converted to Python floating-point numbers. If the WebFOCUS COMPUTE defines the returned field as integer, the decimal point and any decimal places will be truncated. In the csvin file, if a non-numeric field contains the double quotation character, it will be doubled by WebFOCUS. Python will correctly parse this because the Dialect.doublequote format parameter of the Python csv module defaults to *True*.

- **Output.** The function can return multiple output fields. However, each WebFOCUS COMPUTE command can only retrieve a single output field. To retrieve multiple output fields, issue multiple COMPUTE commands. In the call to the PYTHON function, the output argument (the last argument) must match the name of a field in the OUTPUT_DATA segment of the synonym generated for the Python script.

  For example, in the following Master File, the output argument is called ADDITION:

  ```
  SEGMENT=OUTPUT_DATA, SEGTYPE=U, PARENT=INPUT_DATA, $
    FIELDNAME=ADDITION, ALIAS=addition, USAGE=D7.1,
          ACTUAL=STRING,      MISSING=ON,      TITLE='Addition', $
  ```

  Therefore, the last (output) argument name in the call to PYTHON must be ADDITION:

  ```
  COMPUTE Anyname/I5 = PYTHON(synonym_name, anyarg1, anyarg2, ADDITION);
  ```

  The output written to csvout must be a sequence, for example, a list, even for a single field.

  For a list containing a single field, the correct syntax is:

  ```
  writer.writerow([result])
  ```

  The following syntax is incorrect and will return incorrect values for strings and raise an exception for numeric fields:

  ```
  writer.writerow(result)
  ```

- **Functions.** The Python script can contain one or more user-defined functions. When the metadata object (synonym) is created for the script, one of these functions must be selected as the starting point for execution of the script. The definition of the user-written function used as the starting point must contain csvin and csvout as the first positional arguments.

  Note: Because the Python script will be imported, the following Python programming idiom will be ignored.

  ```
  if __name__ == '__main__':
  ```

  However, including it may be useful for testing outside of WebFOCUS.

- **Headers.** Using a header record listing the field names (instead of using positional index numbers) is not required in the sample input data file when creating the synonym for the Python script or when sending data to and retrieving data from the Python script. However, using header records, and, therefore, field names, in the Python script makes it more readable. The following syntax shows a sample of how to implement headers. In the csv.DictReader statement, use of a header record is implied:

  ```
  fieldnames = ['addition']
          reader = csv.DictReader(file_in, quoting=csv.QUOTE_NONNUMERIC)
          writer = csv.DictWriter(file_out, quoting=csv.QUOTE_NONNUMERIC,
                                  fieldnames=fieldnames)
          writer.writeheader()
  ```

  The recommendation is to use header records in the input and output .csv files. If you use sample data without a header, the field names in the generated metadata will be of the form FIELD_1 through FIELD_*n*.

The Adapter for Python also comes with a set or predefined statistical Python functions that you can easily invoke in WebFOCUS.

Prerequisites for Using the Adapter for Python

You can access the complete list of prerequisites when you configure the adapter by right-clicking the adapter name and clicking *Prerequisites*.

On the Linux x64 Intel and Windows platforms, the server includes a fully configured Python 3.6.x release (with required packages) in the EDAHOME etc/python directory. Therefore, a separate Python download, installation, and package installation steps are not required. For these platforms, use the full path EDAHOME etc/python directory displayed in the sample text.

While the EDAHOME etc/python release is the preferred configuration choice, you can still point an external release, if needed (such as for additional packages), provided it conforms to the instructions listed in the Prerequisites page for the adapter.

- Python must be installed on the same machine as the WebFOCUS Reporting Server, using the same bit size (32 or 64-bit).
- The Adapter for Python is available on Windows, z/OS, and Linux.
- The required Python release level is 3.6.x. Do not deselect the pip option in the install.
- The following packages must be installed in the following order (instructions are on the Prerequisites page):
  - numpy.
  - scipy (needs numpy).
  - scikit-learn (needs both of them and will add additional packages).
  - pandas.
- The system variables you may need to set are described on the Prerequisites page for the adapter.

Configuring the Adapter for Python

Configuring the adapter consists of identifying your Python installation directory.

Procedure: How to Configure the Adapter for Python

1. In the Reporting Server Web Console, click *Connect to Data*.
2. Click *New Datasource*, the right-click *PYTHON*. Note that PYTHON can be found in the Statistics category on the *Available* drop-down list.

   The Add PYTHON to Configuration page opens.
3. Enter the path to your Python installation directory, and click *Test*.

   The following message displays for a successful configuration.

   ```
   Successful test for the Python environment
   ```
4. Click *Configure*

   The Adapter for Python is added to the list of Configured Adapters.

Creating Synonyms for Python Functions

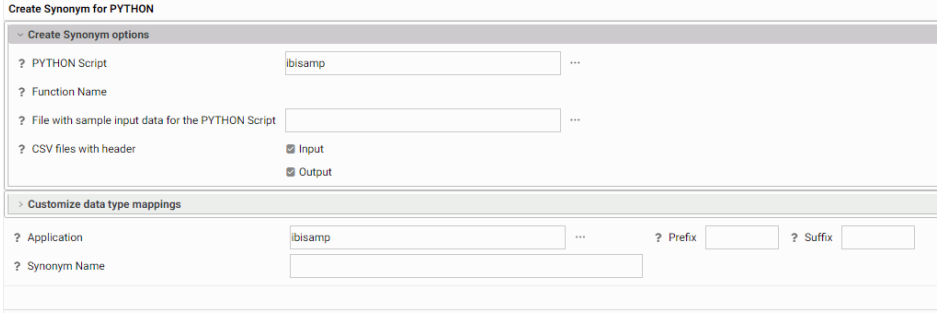
Each Python script used with the Adapter for Python must have a synonym (metadata object) that describes the input fields and output fields of the script. If a Python script contains multiple user written functions, and you want to be able to use more than one function within the script as a starting point, you must create a separate synonym for each function within the script.

The synonym will be created using a sample file that contains only the fields that are input parameters for the script. A few rows of sample data are sufficient for the Adapter for Python to determine the appropriate data types and lengths of the parameters. The sample file must be a .csv file. The data in the file does not have to contain actual data, but it should represent the highest values for numeric fields and the longest lengths for alphanumeric values that will appear the actual data. The Master File will contain the list of input fields and output fields. The Access File will contain information about the script file and sample input file.

Procedure: How to Create a Synonym for a Python Function

1. Right-click PYTHON on the list of configured adapters, and click *Create metadata objects* on the context menu.

   The Create Synonym for Python frame opens, as shown in the following image.



   Note that a metadata object is a synonym. The synonym for a Python function will consist of a Master File (which describes the input fields and output fields needed for running the function) and an Access File (which contains information about the sample data file and the script file).

2. Enter or select values for the following parameters.

   PYTHON Script

   Is the Python script. Enter an application directory name and script name, or click the ellipsis (...) to navigate to an application directory and select a script, then click *OK*. The Python script will have the extension .py.

Function Name

Select the name of the (starting) function in the script file for which to create a synonym.

For example, the Python script named arithmetic_example.py contains the definition for the function adder:

```
def adder(csvin,csvout):
```

Select file with sample input data for the PYTHON Script

Open the file picker (...) to select the application directory and file that contains the sample data for creating the synonym. Click *OK*.

This file is used to determine the field names, data types, and lengths for the data sent to the Python script in *csvin*. If the sample file has no header record, the field names will be FIELD_1 through FIELD_*n.*

CSV files with header

If the input .csv file does not have a header row, uncheck *Input*. If the output .csv file should not have a header row, uncheck *Output*. The header requirements are contained in the function code.

For example, in the following sample code the input file contains no header record (fieldnames) and the fieldnames object is defined at runtime using the fieldnames argument for the reader. The output file will contain a header record, as defined in the fieldnames argument for the writer, and is written to the file using the statement writer.writeheader():

```
with open(csvin,  'r', newline='') as file_in, \
         open(csvout, 'w', newline='') as file_out:
reader = csv.DictReader(file_in, fieldnames=['input_field'],
                        quoting=csv.QUOTE_NONNUMERIC)
writer = csv.DictWriter(file_out, fieldnames=['output_field'],
                        quoting=csv.QUOTE_NONNUMERIC)
writer.writeheader()
```

Application

Enter the name of the application directory in which to create the synonym, or click the ellipsis (...) to navigate to an application directory, then click *OK*.

Synonym Name

Enter a name for the resulting synonym, or accept the default name.

3. Click the Create Synonym button on the ribbon.

The synonym is created in the specified application directory.

Note: You can generate sample Python scripts and data files in the Reporting Server Web Console. Create an application directory to contain the sample files, right-click the application folder, point to *New*, and click *Tutorials*. On the Tutorials page, select the *WebFOCUS - Retail Demo* tutorial, make sure *Create Python Example* is checked and that *Large* or *Medium* is selected for *Tutorial Data Volume Limit*. Click *Create* to create the demo files.

Running a User-Written Python Script

To run a Python script, you call the WebFOCUS PYTHON function. The arguments you supply to the PYTHON function consist of the synonym for the Python script, the input fields, and the output field.

Syntax: How to Run a User-Written Python Function

```
PYTHON([app/]synonym, input1 [, input2 ...], output)
```

where:

[app/]synonym
Is the application and synonym name for the Python script.

input1 [, input2 ...]
Are the input arguments.

output
Is the output argument. This argument must match the name of a field in the OUTPUT_DATA segment in the Master File.

Example: Running a Python Script

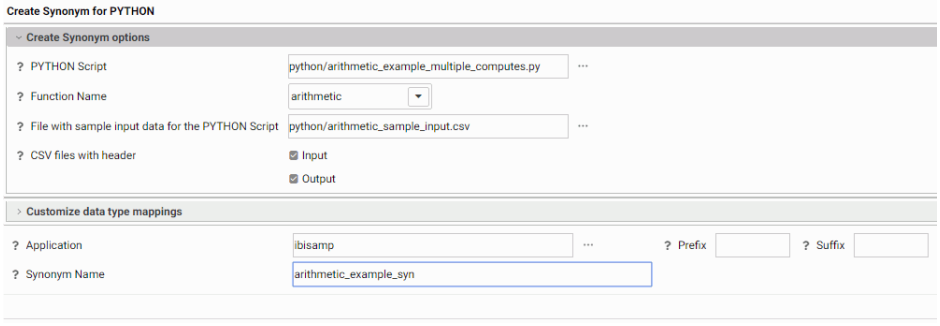
The following is the arithmetic_example_multiple_computes.py Python script, which calculates four output fields, ADDITION, SUBTRACTION, MULTIPLICATION, and DIVISION in the function named *arithmetic*. All of the files for this example reside in an application directory named *python*.

```
# arithmetic_example_multiple_computes.py

import csv
import time

def arithmetic(csvin, csvout):

    with open(csvin,  'r', newline='') as file_in,\
         open(csvout, 'w', newline='') as file_out:

        fieldnames = ['addition', 'subtraction',
                      'multiplication', 'division']

        reader = csv.DictReader(file_in, quoting=csv.QUOTE_NONNUMERIC)

        writer = csv.DictWriter(file_out, quoting=csv.QUOTE_NONNUMERIC,
                                fieldnames=fieldnames)
        writer.writeheader()

        for row in reader:
            addition       = row['a_number'] + row['another_number']
            subtraction    = row['a_number'] - row['another_number']
            multiplication = row['a_number'] * row['another_number']
            division       = row['a_number'] / row['another_number']

            writer.writerow({'addition':       addition,
                             'subtraction':    subtraction,
                             'multiplication': multiplication,
                             'division':       division})
```

The .csv file with the sample data, arithmetic_sample_input.csv, has a header record and two data records to be used to determine the data types and lengths for the input arguments.

```
"a_number","another_number"
1,1
100000,100000
```

The synonym creation frame for this script is shown in the following image.



The generated Master File (arithmetic_example_syn.mas) follows:

```
FILENAME=ARITHMETIC_EXAMPLE_SYN, SUFFIX=PYTHON  , $
  SEGMENT=INPUT_DATA, SEGTYPE=S0, $
    FIELDNAME=A_NUMBER, ALIAS=a_number, USAGE=I11, ACTUAL=STRING,
      MISSING=ON,
      TITLE='a_number', $
    FIELDNAME=ANOTHER_NUMBER, ALIAS=another_number, USAGE=I11,
ACTUAL=STRING,
      MISSING=ON,
      TITLE='another_number', $
  SEGMENT=OUTPUT_DATA, SEGTYPE=U, PARENT=INPUT_DATA, $
    FIELDNAME=ADDITION, ALIAS=addition, USAGE=D10.1, ACTUAL=STRING,
      MISSING=ON,
      TITLE='addition', $
    FIELDNAME=SUBTRACTION, ALIAS=subtraction, USAGE=D5.1, ACTUAL=STRING,
      MISSING=ON,
      TITLE='subtraction', $
    FIELDNAME=MULTIPLICATION, ALIAS=multiplication, USAGE=D15.1,
ACTUAL=STRING,
      MISSING=ON,
      TITLE='multiplication', $
    FIELDNAME=DIVISION, ALIAS=division, USAGE=D5.1, ACTUAL=STRING,
      MISSING=ON,
      TITLE='division', $
```

The generated Access File (arithmetic_example_syn.acx) follows:

```
SEGNAME=INPUT_DATA,
  MODNAME=python/arithmetic_example_multiple_computes.py,
  FUNCTION=arithmetic,
  PYTHON_INPUT_SAMPL=python/arithmetic_sample_input.csv,
  INPUT_HEADER=YES,
  OUTPUT_HEADER=YES, $
```

The following WebFOCUS procedure, sales_multiple_computes.fex calls the arithmetic function four times in order to get a value returned for each of the four outputs:

```
-* sales_multiple_computes.fex

TABLE FILE GGSALES
SUM
  DOLLARS
  UNITS

COMPUTE Addition/D7        = PYTHON(python/arithmetic_example_syn,
                              DOLLARS, UNITS, ADDITION);
COMPUTE Subtraction/D7     = PYTHON(python/arithmetic_example_syn,
                              DOLLARS, UNITS, SUBTRACTION);
COMPUTE Multiplication/D16 = PYTHON(python/arithmetic_example_syn,
                              DOLLARS, UNITS, MULTIPLICATION);
COMPUTE Division/D7.2      = PYTHON(python/arithmetic_example_syn,
                              DOLLARS, UNITS, DIVISION);

WHERE RECORDLIMIT EQ  100
HEADING
  "Arithmetic Example, Multiple Computes"
  ""

ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image.

Arithmetic Example, Multiple Computes

Dollar Sales	Unit Sales	Addition	Subtraction	Multiplication	Division
1343927	105860	1,449,787	1,238,067	142,268,112,220	12.70

Chapter 83

Using the Adapter for Query/400

The Adapter for Query/400 enables you to create synonyms that WebFOCUS can use for reporting against a Query/400 query definition.

The adapter generates metadata from an existing Query/400 object in such a way that only the required fields and ancillary information is retained for use as a basis for requests or queries.

In this chapter:

Preparing the Adapter for Query/400 Environment

The Adapter for Query/400 does not require setting any environment variables.

However, any query definitions for which you wish to create synonyms must already be available to the adapter. You cannot use the adapter to create these definitions.

Configuring the Adapter for Query/400

You can configure the Adapter for Query/400 from the Web Console or from the Data Management Console.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Managing Query/400 Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of an existing Query/400 request and its underlying tables, columns, and associated data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Query/400 request that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.

- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Query/400

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Library Name

Supply the name of the library in which the query definitions reside. (Wildcards are not permitted.)

Note: When you create a synonym for Query/400 on the IBM i platform, standard IBM i naming conventions apply to the target data source. Therefore, the Adapter for Query/400 supports the use of double-quotation marks around any library name and/or file name that contains lower case or NLS characters.

Click *Submit* to continue.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Do not save library information

Select this check box if you do not wish to save the library of the stored query in the metadata.

This selection ensures that the specific execution is based on the current library path of the connected user at run time.

Note that the library referred to here is *not* the library specification of the files used within the query, but rather where the query itself is located.

Select objects for synonym creation

To select all names in the list, select the check box to the left of the *Default Synonym Name* column heading. (If the library contains a large number of query definitions, this check box will not appear, however, synonyms will be created for all definitions automatically.)

To select specific names, click the corresponding check boxes.

Not all listed objects are data-related. Be sure to select those that are appropriate for creating synonyms.

Note: Mixed case names or names with NLS character will appear in double quotation marks. However, the double quotation marks will be replaced by underscore characters in the Default Synonym Name column (see below).

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Note: If you see a synonym name surrounded by underscore characters (as described in the previous note), you can remove the underscores if you wish.

Example: Creating a Synonym for Query/400

To generate the following synonym from the Create Synonym panes:

1. Specify a library in the Library Name field, then click *Submit*.
2. On the second Create Synonym pane, choose a query from the Synonym list and click *Create Synonym*.

   The synonym is created and added under the specified application directory (the default is baseapp).

   A status window displays the message: *Created Successfully*
3. From the message window, click *Applications* on the menu bar.
4. Open the *baseapp* application folder in the navigation pane and click the synonym.
5. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

   Sample Generated Master File:

```
FILENAME=CUSTSUM, SUFFIX=QRYI    , $
  SEGMENT=TMPFILE, SEGTYPE=S0, $
    FIELDNAME=YEAR, ALIAS=YEAR, USAGE=I9, ACTUAL=I4,
     TITLE='YEAR', $
    FIELDNAME=MONTH, ALIAS=MONTH, USAGE=I9, ACTUAL=I4,
      TITLE='MONTH', $
    FIELDNAME=DAY, ALIAS=DAY, USAGE=I9, ACTUAL=I4,
      TITLE='DAY', $
    FIELDNAME=CUSTOMER, ALIAS=CUSTOMER, USAGE=A25, ACTUAL=A25,
      TITLE='CUSTOMER', $
    FIELDNAME=QUANTITY, ALIAS=QUANTITY, USAGE=P15.2C, ACTUAL=P8,
      TITLE='QUANTITY', $
    FIELDNAME=REVENUE, ALIAS=REVENUE_WO_TAX, USAGE=P15.2C, ACTUAL=P8,
      TITLE='REVENUE_WO_TAX', $
    FIELDNAME=CUSTSUM, ALIAS=FILE1, USAGE=A32, ACTUAL=A32,
      ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='ACCT/CUSTOMER', $
```

   Sample Generated Access File:

```
SEGNAME=TMPFILE, QIQRY=ACCT/CUSTSUM, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Chapter 84

Using the Adapter for Rdb

The Adapter for Rdb allows applications to access Rdb data sources. The adapter converts application requests into native Rdb statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Rdb Environment

Oracle Rdb allows two types of installation:

- **Standard.** Supports only one Rdb release level on a machine. No preparation is required to use the adapter in a Standard Rdb environment.
- **Multi Version.** Supports multiple Rdb release levels on the same machine. To switch between releases, Multi Version supplies a script, which sets up logically to point to standard names, such as SQL$, at specific release files, such as SQL$61.EXE, SQL $70EXE, SQL$71.EXE, and SQL$72.EXE.

To use the adapter in a Multi Version environment, prior to server startup, your Rdb environment must be set up to access the database file release level that corresponds to the release level selected during configuration of the adapter. To enable switching between Rdb releases, your site might use the Rdb script DECRDB$SETVER, a site-specific script, or a symbol. For details on how Rdb release level switching is implemented at your site, see your OpenVMS Administrator.

Configuring the Adapter for Rdb

Configuring the adapter consists of specifying connection and authentication information for the connection you want to establish.

To connect to an Rdb database server, the adapter requires connection and authentication information. You supply this information using the SET SERVER command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Rdb

The Rdb adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Server

Full path to an Rdb data source, or a logical name, such as SQL$DATABASE, which contains the full path. Rdb limits this value to 31 characters. If the full path exceeds this limit, you must specify a logical name to point to the full path.

- **Full Path.** Full path specification must be in the following form: *device_name*:[*directory*]*file*.RDB. The RDB extension is required.
- **Logical Name.** To use this method, the logical must be set prior to server startup. The trailing blank is optional.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Note: Only one SET SERVER command may be active at any given moment; the last command issued is the active Rdb DBMS file.

Rdb multischema DBMS files are not supported for metadata creation purposes, but are valid for data access purposes provided that the Access File TABLENAME= *value* points to the table properly. For more information about configuration and metadata creation, see *Using Multischema Rdb DBMS Files* on page 1995.

Syntax: How to Declare Connection Attributes Manually

The adapter connects to Rdb using the credentials of the operating system user impersonated by the data access agent.

```
ENGINE SQLRDB SET SERVER server
```

where:

SQLRDB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

server

Is the full path to an Rdb data source, or a logical name, such as SQL$DATABASE, which contains the full path. Rdb limits this value to 31 characters. If the full path exceeds this limit, you must specify a logical name to point to the full path.

Full Path. Full path specification must be in the following form: *device_name*:[*directory*]*file*.RDB. The RDB extension is required.

Logical Name. To use this method, the logical must be set prior to server startup. The trailing blank is optional.

Note: Once the initial SET SERVER command is saved in a profile, you can manually declare additional connections to more than one Rdb database by including any number of additional Rdb ATTACH ALIAS commands and editing the Access File to properly point to the TABLENAME= *value* at the alias. For more information about configuration and metadata creation in this context, see *Using Multischema Rdb DBMS Files* on page 1995.

Example: Declaring a Connection to ORDERS.RDB

The following example shows how to declare a connection to the ORDERS.RDB data source, located in the OUTBOUND directory on disk.

```
ENGINE SQLRDB SET SERVER DISK$SHIPPING:[OUTBOUND]ORDERS.RDB
```

Overriding the Default Connection

Once connections have been defined, the connection named in the last SET SERVER command serves as the default connection. You can override this default by reissuing the SET SERVER command.

Syntax: How to Change the Default Connection

```
ENGINE SQLRDB SET DEFAULT_CONNECTION connection
```

where:

SQLRDB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLRDB SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLRDB SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

SQLRDB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Rdb Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Rdb data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Rdb table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Rdb

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Rdb Data Type Support* on page 1990.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Rdb synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLRDB ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,'TABLENAME=RDB$HANDLE.NF29004',
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Rdb table.
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Rdb Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Using Multiple Rdb DBMS Files

In native Rdb, within the same session you can access two or more tables that exist in different physical Rdb DBMS files by issuing one or more ATTACH ALIAS commands using the following syntax,

```
ATTACH 'ALIAS alias FILENAME filename'
```

and referencing a specific ALIAS or RDB$HANDLE for the current default file in each SQL request you issue.

Although you can accomplish the same task using the Adapter for the Rdb DBMS, the adapter metadata-creation facility is limited to the single default connection handle. Therefore, if you wish to use multiple Rdb tables, you must add one or more SQL Passthru ATTACH commands to the server profile (or a group or user profile), and complete a few extra metadata-creation steps, as described in *Completing Setup and Metadata Creation Steps for Multiple Rdb DBMS Files* on page 1991.

If you follow these instructions, you will be able to JOIN across Rdb DBMS files and access Rdb data remotely on a subserver from a client connection (using a SUFFIX=EDA synonym).

The details of the extra steps for metadata creation will vary by site, however, two general approaches are available:

- If a single Rdb DBMS file with a common set of tables is available as a dictionary, you can configure this Rdb file as the default connection. The steps that follow are as simple as the standard synonym creation process, with the addition of a manual edit to the resulting Access File to point the TABLENAME=*value* at the desired ATTACH alias. *Completing Setup and Metadata Creation Steps for Multiple Rdb DBMS Files* on page 1991 illustrates this process.
- If an Rdb DBMS file is not available as a single dictionary, after configuring an initial connection, creating metadata, and editing the Access File TABLENAME= attribute, you can reconfigure and repeat the synonym creation and Access File editing steps as many times as necessary for the additional Rdb targets.

Important: The Adapter for Rdb has been stabilized. Therefore, Information Builders has no plans to support multiple connections directly through the adapter, except by using the configuration techniques described in this topic.

Example: Completing Setup and Metadata Creation Steps for Multiple Rdb DBMS Files

This example illustrates the setup and metadata creation steps for multiple Rdb DBMS files. The example assumes that an Rdb DBMS file with all tables is available as a template dictionary and for use in metadata creation. You can adapt this example to suit your needs.

1. Determine what Rdb files are needed. For this example, the file names are:

   ```
   CORPTABLES.RDB,

   CORPDATA2006.RDB

   CORPDATA2007.RDB

   CORPDATA2008.RDB
   ```

   These files reside in the directory DISK$DUA0:[RDB].

   CORPTABLES is an empty DBMS template that is cloned for each year's data and provides a dictionary for use in the synonym creation process. In addition, logicals that match the physical file names are used to avoid the Rdb 31-character physical limit and to facilitate coding in case the location changes in the future.

2. To ensure that the logicals are always available (and known to the server at start up time), edit the following file

   ```
   EDACONF [.BIN]EDAENV.COM
   ```

and add a logical name reference (of 31 characters or less) for each Rdb DBMS file to be accessed. For example:

```
$ DEFINE CORPTABLES   DISK$DUA0:[RDB]CORPTABLES.RDB

$ DEFINE CORPDATA2006 DISK$DUA0:[RDB]CORPDATA2006.RDB

$ DEFINE CORPDATA2007 DISK$DUA0:[RDB]CORPDATA2007.RDB

$ DEFINE CORPDATA2008 DISK$DUA0:[RDB]CORPDATA2008.RDB
```

Save your changes.

Note:

- If a generic application setup batch file with the desired logical is available and does not force a process exit, you can call that batch file from EDAENV.COM and avoid the individual entries described above.
- If the desired logicals are already available, you can skip this step altogether because the logicals are automatically issued when you boot your machine.

3. Assuming you have edited the EDAENV file as described in step 2, start or restart your server.
4. During Rdb adapter configuration, choose the standard options for enabling Rdb data access and supply the following logical name for the server:

   ```
   CORPTABLES:
   ```

   The resulting profile entry would looks as follows:

   ```
   ENGINE SQLRDB SET SERVER CORPTABLES:
   ```

5. Using the Web Console's edit facility or another editor, modify the global profile, edasprof.prf (or possibly a group or user profile if you are configuring for group or personal profile-based behavior) to add an ATTACH ALIAS command for each of the additional Rdb files. This step will make the Rdb database files visible as aliases to the default database. For this example, the following code was added after the SET SERVER line in the profile:

   ```
   ENGINE SQLRDB SET SERVER CORPTABLES:

   SQL RDB ATTACH 'ALIAS CORPDATA2006 FILENAME CORPDATA2006:' ;
   END
   SQL RDB ATTACH 'ALIAS CORPDATA2007 FILENAME CORPDATA2007:' ;
   END
   SQL RDB ATTACH 'ALIAS CORPDATA2008 FILENAME CORPDATA2008:' ;
   END
   ```

Note:

- The semi-colon and END statements are required.
- The ATTACH statements must follow the initial SET SERVER command. (Placement before the SET SERVER command will cause a crash.)
- You can issue additional SET SERVER statements after the ATTACH statements.

6. At this point, if you have metadata from a prior installation it is good practice to determine if you can use your existing metadata as is, if you need to modify it, or if you need to create new (additional) metadata that is distinct from what already exists. Depending on your needs, you may be done or you may need to continue with either Step 7 or Step 8.

   - For new installations (that is, those with no prior metadata), continue with Step 7.
   - In releases prior to 7.6.8, a synonym was either prefixed or not prefixed depending on how the metadata was created or, possibly, edited at a later time for a specific value. If the metadata for a given table is available and the TABLENAME=*value* attribute in the Access File points at the appropriate SQL *alias.tablename* (or is not prefixed for data that uses the default connection), then there is no need to modify the given synonym in order to access the data. If all synonyms are of this type and no additional new metadata is desired, you are done.
   - If additional new synonyms are desired, or if prefixes on existing synonyms need to change, continue with Step 7 to create the new metadata and then go to Step 8 to edit the Access File TABLENAME=*value* for the existing and new synonyms.

7. Whether you are creating a synonym for the first time or re-creating a synonym that existed in a prior release, you can use the Web Console or the DMC synonym creation facility. When prompted, select the desired Rdb tables using the connection:

   ```
   CORPTABLES:
   ```

8. After any needed metadata has been created, edit the Access Files that needs to change (using the Web Console editor or another editor) to have a TABLENAME= *value* that is prefixed with the appropriate ALIAS reference.

   To complete this step from the Web Console, locate the synonym whose Access File you need to edit, click it, and select *Edit Access File as Text* from the menu.

   For instance, in the Access File TABLENAME entry for the 2007 Sales table initially looks like this:

   ```
   TABLENAME='RDB$HANDLE.SALES',
   ```

   The single quotation marks are required since the $ in RDB$HANDLE is regarded as a special character.

The edited version would look as follows:

```
TABLENAME='CORPDATA2007.SALES',
```

In this instance, the single quotation marks are optional since CORPDATA2007 does not contain a special character.

9. Repeat step 8, as many time as necessary to account for all metadata whose access requires an ATTACH ALIAS command.

   Tip: As an alternative to editing each file individually, you can create a batch script that edits the RDB$HANDLE string for groups of files at one time.

Note:

- If you prefer to use a different default Rdb file rather than the current one, you can switch the configuration value to the preferred value once metadata creation is done. To do so, return to the Rdb adapter configuration pane and change the Rdb target. Subsequently, any Access File using an explicit ATTACH ALIAS prefix, the RDB$HANDLE prefix, or no prefix at all will use the associated file as the source of data.
- If a single Rdb DBMS file is not available as a template for all tables, you can follow the basic configuration and synonym creation process, however, you will need to reconfigure the Rdb target, then repeat Step 7 (metadata creation) for each individual Rdb target, and continue to Step 8 (Access File editing).

Tip: An orderly approach is to create all of the synonyms, repeat the reconfiguration step as needed, then edit the Access Files.

Reference: Considerations For Sites That Have Upgraded From a Release Prior to 768

Releases prior to 768 recommended similar metadata creation and Access File editing steps (with TABLENAME= *prefix.table* and a SERVER= *prefix value*) for JOINs, and, in addition, required that the profile be edited to eliminate the SET SERVER command and to include the SQL APT=OFF command at runtime.

While not explicitly documented, this method enabled remote access (using SUFFIX=EDA) to Rdb data because APT=OFF allowed Rdb to pick up data using the SERVER= *value* from the Access File.

While it is not necessary for sites that currently use this configuration method to change or reconfigure in Version 7 Release 6.8, there are distinct advantages in doing so:

- When the SET SERVER command was removed, as required under the prior configuration method, the Web Console synonym creation capability was disabled. Since It is no longer necessary to remove the SET SERVER command, keeping the command will enable you can to create synonyms for Rdb from the Web Console without any secondary configurations.
- The need to remove the SET SERVER command and turn APT to OFF may have provided sufficient justification for configuring a secondary server to be used exclusively for JOIN and remote (SUFFIX=EDA) purposes under the prior method. Under the new method, these reasons have been eliminated. Therefore, you may wish to consider eliminating the secondary server in order to save maintenance overhead and computer cycles. Using the explicit ATTACH ALIAS steps described in this document also provides clearer information about how data is being accessed.

Note that it is not necessary to remove the SERVER= *value* from the Access Files since this information is not used in this context and is, therefore, ignored.

Using Multischema Rdb DBMS Files

A multischema Rdb DBMS file is, in effect, a file with multiple schemas within one physical file and, as such, two part names in a non-multischema file become three part names (that is, the two part name *schema.table,* where ATTACH ALIAS assigns the schema name, becomes *catalog.schema.table*). As a result, you can ATTACH a single file (rather than a default plus various aliases).

The Adapter for Rdb does not support multischema Rdb DBMS files for metadata creation. However, if a non-multischema Rdb file is available as a template for tables, synonym creation and data access may be done in much the same way as described previously for multiple Rdb DBMS files (see *Completing Setup and Metadata Creation Steps for Multiple Rdb DBMS Files* on page 1991), with the following variations.

For a multischema Rdb DBMS file, you can use either of the following commands for data access at run time

```
ENGINE SQLRDB SET SERVER multischema_target;
```

or, as a non-aliased ATTACH command,

```
SQL RDB ATTACH 'FILENAME multischema_target';
END
```

With either of these commands the proper TABLENAME= *value* is rendered in the Access File as a three part name—for example,

```
TABLENAME=ADMINISTRATION.PERSONNEL.EMPLOYEES,
```

which is a name from Rdb's standard multischema example used in non alias mode.

For an explicitly aliased multischema ATTACH command, use the following method. A multischema Rdb DBMS file used with an ATTACH ALIAS command requires (as Rdb's SQL$ requires) that you add a SET QUOTING RULES command to ensure that the execution process for the table is properly referenced. The easiest way to do this is to add the command in the same place as the ATTACH ALIAS command. (Note that you can place the QUOTING RULES command before or after the ATTACH command, but it must be executed before actual data access.)

For example (also using Rdb's multischema example, but in alias mode), add the following code:

```
SET QUOTING RULES 'SQL92';
END
SQL RDB ATTACH 'ALIAS CORP FILENAME CORPORATE_DATA:' ;
END
```

A proper TABLENAME= *value* is a specially quoted three part name such as:

```
TABLENAME='"CORP.ADMINISTRATION".PERSONNEL.EMPLOYEES',
```

Note the use of enclosing single quotation marks: these are required because the double quotation marks are considered special characters to be passed to Rdb, which in turn processes the "CORP.ADMINISTRATION" portion properly as the alias due to Rdb's SET QUOTING RULES command, which is used in the profile.

Rdb Database Driver Performance

The default of Rdb is to open tables in read/write mode unless told otherwise. The Server supports read/write operations, so this is appropriate behavior. However, Rdb read/write operations consume more resources and are slower, even if just a select is being done. Applications can set Rdb opens to Read-only on a per request basis or a per session basis to enhance resource usage and speed performance. In either case, a commit should be done to ensure that prior work is complete.

On a session basis, add the following to the EDASPROF.PRF after the SQL SQLRDB SET SERVER command:

```
SQL SQLRDB COMMIT ;
SQL SQLRDB SET DECLARE TRANSACTION READ ONLY ;
```

On a per request basis (for example, immediately before a TABLE request) the usage is:

```
SQL SQLRDB COMMIT ;
SQL SQLRDB SET TRANSACTION READ ONLY ;
```

Chapter 85

Using the Adapter for Remote Servers

The Adapter for Remote Servers allows applications to access data sources that reside on a remote server.

Using a mechanism called platform transparency, you can join data sources that reside on different platforms. The communication protocols used to connect to the servers may be the same or different.

In this chapter:

Configuring Remote Servers

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Remote Servers

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Basic

NODE

8-character string

Defines the logical name of a node block, which must be unique in the file. The logical name can be a maximum of 8 characters, must begin with a letter, and can include any characters, except semicolon and equal sign. A node block contains keyword-value pairs, which are enclosed in a BEGIN-END statement. The following is an example of a node block:

```
NODE = ABC
BEGIN
   DESCRIPTION = client node block
   CLASS = CLIENT
   PROTOCOL = TCP
   HOST = localhost
   PORT = 8100
END
```

HOST

string

Defines the host that a client is connecting to or an IP address that a listener is listening on.

PORT

positive integer number

Defines the TCP port number that a client is connecting to or a listener is listening on.

HTTP_PORT

positive integer number

Defines the TCP port number on which the remote server's Web Console listener is listening. Also used in declaring a communication block for the purpose of Remote Synonyms.

CLASS

If this is a z/OS server, you must include a qualifier.

SECURITY

Defines the authentication method used in communication to a remote server if no credentials are provided to the application.

- *IWA* for Integrated Windows Authentication between a Windows client and a Windows server.
- *PWP* for password passthrough authentication.
- *TRUSTED* for a trusted connection, where the current logged on user ID is passed to the server.
- *userid,password* where any value other than IWA, PWP or TRUSTED is considered a userid and password string for EXPLICIT authentication.

  The information may be clear text or encrypted. Encrypting is automatic when the Web Console is used to set values.

  Any ENGINE EDA SET CONNECTION_ATTRIBUTES *server/userid,password* command issued in a profile (server, group or user) or in a procedure overrides any value set by SECURITY = (including IWA, PWP, TRUSTED).

DESCRIPTION

string

Description for the server node.

Advanced. These options enable you to provide connection information for specific services.

SERVICE NAME

CLIENT (servicename)

Defines how to send outbound communications to a remote server.

servicename is optional. If provided, it must match the value of SERVICE in the service block of the server.

HTTP_SSL

0 or 1

Defines whether the Secure Sockets Layer protocol is used in the remote server Web Console listener:

- *0* if no SSL is used in the connection to the Web Console.
- *1* if SSL is used in the connection to the Web Console.

COMPRESSION

0 or 1

Activates data compression in a data transfer between client and server:

- *0* for no compression.
- *1* for compression on.

ENCRYPTION

0 or DES or Advanced or IBCRYPT

Defines the encryption method used in data transfer between client and server:

- *0* for no encryption.
- *DES* for 56-bit fixed-key Data Encryption Standard.
- *Advanced,* enables you to more easily select and combine ciphers, modes, and RSA key lengths. This option provides the following ciphers: 3DES, AES 128, AES192, AES 256; and the following modes: ECB and CBC.
- *IBCRYPT* for user-defined algorithm. The key is 512-bit RSA-encrypted.

Note: Only 0 and DES are supported for HTTP protocol.

CONNECT_LIMIT

number of seconds

Defines the maximum time that the client will wait for a TCP connection response from the server:

- -1 for indefinite wait.
- 0 for no wait.

MAXWAIT

Defines the time, in seconds, that the client will wait for a response from the server:

-1 indicates an indefinite timeout.

Managing Metadata for Remote Servers

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each metadata description that is managed by the subserver. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access file that represent the server metadata.

You can use the Adapter for Remote Servers to create synonyms for remote data sources that reside on a subserver. If the data source is remote, the Master File contains the attribute SUFFIX=EDA.

Note: The Adapter for Remote Servers defines a relational data source, so its Master Files are single-segment. When a synonym is created from a multi-segment hierarchical table, like the CAR file, all its segments will be flattened out into one segment in the resulting Master File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Remote Servers

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): tables, views, and any other supported objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

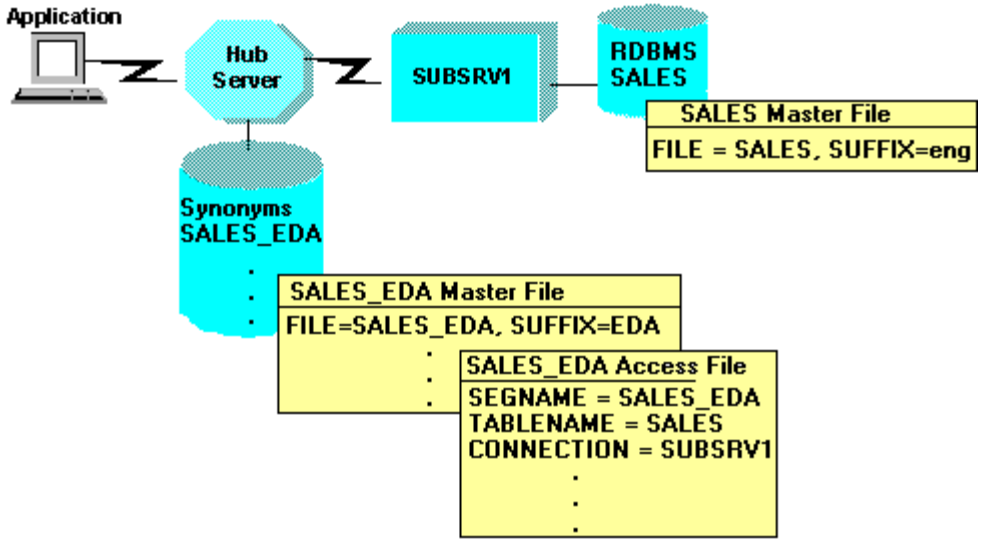
Example: Synonym for a Remote Data Source

The following diagram illustrates the use of Master Files and Access Files for data sources. A synonym called SALES_EDA was created on a hub server using a SALES file located on a subserver.

- The multiple segments of the SALES file are flattened out in the SALES_EDA Master File, which is now a single-segment table.
- The SALES_EDA Master File contains SUFFIX=EDA to indicate that the data source is remote.
- The SALE_EDA Access File includes the synonym for the data source, the name of the data source, and the location of the data source:

```
SEGNAME = SALES_EDA
TABLENAME = SALES
CONNECTION = SUBSRV1
```

When you connect to the Hub Server, the SALES_EDA Master File and Access File are used to access the remote data source SALES, located on SUBSRV1.



In the above Access File example, the value SUBSRV1 of the CONNECTION parameter must match an outbound node block in the Hub Server communications file.

Here's the original synonym on the sub-server:

Sales.mas

```
FILENAME=KSALES, SUFFIX=FOC     ,                $
  SEGMENT=STOR_SEG, SEGTYPE=S1,                   $
    FIELDNAME=STORE_CODE, ALIAS=SNO, USAGE=A3,    $
    FIELDNAME=CITY, ALIAS=CTY, USAGE=A15,         $
    FIELDNAME=AREA, ALIAS=LOC, USAGE=A1,          $
$ $
  SEGMENT=DATE_SEG, SEGTYPE=SH1, PARENT=STOR_SEG, $
    FIELDNAME=DATE, ALIAS=DTE, USAGE=A4MD, $
$ $
  SEGMENT=PRODUCT, SEGTYPE=S1, PARENT=DATE_SEG,                $
    FIELDNAME=PROD_CODE, ALIAS=PCODE, USAGE=A3, FIELDTYPE=I,  $
    FIELDNAME=UNIT_SOLD, ALIAS=SOLD, USAGE=I5,                $
    FIELDNAME=RETAIL_PRICE, ALIAS=RP, USAGE=D5.2M,            $
    FIELDNAME=DELIVER_AMT, ALIAS=SHIP, USAGE=I5,              $
    FIELDNAME=OPENING_AMT, ALIAS=INV, USAGE=I5,               $
    FIELDNAME=RETURNS, ALIAS=RTN, USAGE=I3,       MISSING=ON, $
  FIELDNAME=DAMAGED, ALIAS=BAD, USAGE=I3,         MISSING=ON, $
```

Here's the synonym that is created by the Create Synonym utility:

Sales_eda.mas

```
FILENAME=SALES_EDA, SUFFIX=EDA     , $
  SEGMENT=SALES_EDA, SEGTYPE=S0, $
   FIELDNAME=STORE_CODE, ALIAS=STORE_CODE, USAGE=A3, ACTUAL=A3, $
   FIELDNAME=DATE, ALIAS=DATE, USAGE=A4MD, ACTUAL=A4MD, $
   FIELDNAME=PROD_CODE, ALIAS=PROD_CODE, USAGE=A3, ACTUAL=A3, $
   FIELDNAME=CITY, ALIAS=CITY, USAGE=A15, ACTUAL=A15, $|
   FIELDNAME=AREA, ALIAS=AREA, USAGE=A1, ACTUAL=A1, $
   FIELDNAME=UNIT_SOLD, ALIAS=UNIT_SOLD, USAGE=I5, ACTUAL=I4, $
   FIELDNAME=RETAIL_PRICE, ALIAS=RETAIL_PRICE, USAGE=D5.2M, ACTUAL=D8, $
   FIELDNAME=DELIVER_AMT, ALIAS=DELIVER_AMT, USAGE=I5, ACTUAL=I4, $
   FIELDNAME=OPENING_AMT, ALIAS=OPENING_AMT, USAGE=I5, ACTUAL=I4, $
   FIELDNAME=RETURNS, ALIAS=RETURNS, USAGE=I3, ACTUAL=I4,  MISSING=ON, $
   FIELDNAME=DAMAGED, ALIAS=DAMAGED, USAGE=I3, ACTUAL=I4,  MISSING=ON, $
```

Sales_eda.acx

```
SEGNAME=SALES_EDA, TABLENAME=EDADBA.sales, CONNECTION=server, KEYS=3, $
```

Notice not only is there no structure, but the three keys (STORE_CODE, DATE and PROD_CODE) were moved to the top.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Access File Keywords

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: TABLENAME=[*location.*][*owner.*]*tablename*

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first *n* fields in the Master File segment.
WRITE	Specifies whether write operations are allowed against the table.

Example: Storing a Server Name in an Access File

The following example shows how to store the server name IBMSERVE in an Access File.

```
SEGNAME=ONE,TABLENAME=CAR,KEYS=1,WRITE=YES,SERVER=IBMSERVE,$
```

Note: Regardless of the type of data source to be accessed, the Access File must contain the following attributes:

- SEGNAME. The segment name in the Access File must match the segment name in the Master File.
- TABLENAME. This attribute specifies the name of the Master File on the server.
- SERVER. This attribute specifies the name of the server.

Example: Using a SUFFIX=EDA Synonym

The Master File of the synonym (DIGIT) on the server is:

```
FILENAME=DIGIT,SUFFIX=FOC,$
 SEGNAME=DIGIT,SEGTYPE=S0,$
  FIELD=THIS_DIGIT ,THIS_DIGIT ,I9    ,I4 ,MISSING=ON,$
  FIELD=SSN        ,SSN        ,A9    ,A9 ,MISSING=ON,$
  FIELD=AMOUNT1    ,AMOUNT1    ,P16.0 ,P8 ,MISSING=ON,$
  FIELD=AMOUNT2    ,AMOUNT2    ,P16.0 ,P8 ,MISSING=ON,$
```

The Master File of the synonym (DIGITEDA) on the client is:

```
FILENAME=DIGITEDA,SUFFIX=EDA,$
 SEGNAME=DIGIT,SEGTYPE=S0,$
  FIELD=THIS_DIGIT ,THIS_DIGIT ,I6   ,I4             ,$
  FIELD=SSN        ,SSN        ,A9   ,A9 ,MISSING=OFF,$
  FIELD=AMOUNT1    ,AMOUNT1    ,P8   ,P8 ,MISSING=ON ,$
  FIELD=AMOUNT2    ,AMOUNT2    ,P9.0 ,P8 ,MISSING=ON ,$
```

The Access File of the synonym (DIGITEDA) on the client points to the file on the server:

```
SEGNAME=DIGIT,TABLENAME=DIGIT,KEYS=1,SERVER=PMSEDA,$
```

where:

- SEGNAME=DIGIT matches the segment name in the server Master File.
- TABLENAME=DIGIT specifies the name of the Master File on the server.
- SERVER=PMSEDA specifies the name of the server.

The EDACS3 client communication configuration file contains a NODE block that points to the server. For example:

```
NAME = EDA CLIENT USING CS/3 TCP/IP
NODE = PMSEDA
 BEGIN
;  TRACE = 31
   PROTOCOL = TCP
   CLASS = CLIENT
   HOST = IBIMVS           ; DNS NAME (PNO 28109)
   SERVICE = 2386          ; TCP/IP PORT FOR SERVER
END
```

The following requests (SQL and TABLE) reference the *local* Master File named DIGITEDA. As shown above, the corresponding Access File contains the server name and the Master File name, as it is known at the server. In this case, Server PMSEDA contains a Master File called DIGIT. At the server, the DIGIT Master File describes a FOCUS data source. The communications configuration file contains an entry for server name PMSEDA so that the FOCUS client can establish communications with the server.

SQL request	TABLE request

`SQL SELECT * FROM DIGITEDA`
`END`
	`TABLE FILE DIGITEDA`
	`PRINT *`
	`END`

Note: The name of an underlying physical file to which a SUFFIX=EDA Access File points must not start with a number. Using a number at the beginning of a file name:

- Will cause an error (FOC14069) when a request attempts to access data using a SUFFIX=EDA synonym that points to a file starting with a number.
- May cause problems when an external application (such as an API application) that does not accept files starting with numbers interacts with a SUFFIX=EDA synonym that points to a file starting with a number.
- May be problematic for a third-party application that does not work with numeric files names or with file names that start with numbers.

Example: Remote Multi-Segment Master and Access File

The following is a multi-segment Synonym. It describes two tables on the sub-server, JOBHIST and JOBLIST that are joined on the column JOBCLASS:

Jobs_eda.mas

```
FILENAME=jobs_eda, SUFFIX=EDA     , $
  SEGMENT=JOBHIST, SEGTYPE=S0, $
    FIELDNAME=PIN, ALIAS=PIN, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=JOBSTART, ALIAS=JOBSTART, USAGE=YMD, ACTUAL=DATE, $
    FIELDNAME=JOBCLASS, ALIAS=JOBCLASS, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=FUNCTITLE, ALIAS=FUNCTITLE, USAGE=A20, ACTUAL=A20, $
  SEGMENT=JOBLIST, SEGTYPE=S0, PARENT=JOBHIST,
    JOIN_WHERE=JOBHIST.JOBCLASS EQ JOBLIST.JOBCLASS;, $
    FIELDNAME=JOBCLASS, ALIAS=JOBCLASS, USAGE=A8, ACTUAL=A8, $
    FIELDNAME=CATEGORY, ALIAS=CATEGORY, USAGE=A25, ACTUAL=A25, $
    FIELDNAME=JOBDESC, ALIAS=JOBDESC, USAGE=A40, ACTUAL=A40, $
    FIELDNAME=LOWSAL, ALIAS=LOWSAL, USAGE=D12.2M, ACTUAL=D8, $
    FIELDNAME=HIGHSAL, ALIAS=HIGHSAL, USAGE=D12.2M, ACTUAL=D8, $
    FIELDNAME=GRADE, ALIAS=GRADE, USAGE=A2, ACTUAL=A2, $
    FIELDNAME=LEVEL, ALIAS=LEVEL, USAGE=A25, ACTUAL=A25, $
```

The following is the corresponding multi-segment Access File:

Jobs_eda.acx

```
SEGNAME=JOBHIST, TABLENAME=EDADBA.jobhist, CONNECTION=server, $
SEGNAME=JOBLIST, TABLENAME=EDADBA.joblist, CONNECTION=server, $
```

Note: While this synonym could not be created using the Web Console Create Synonym feature since it points to two different tables, it could be created in the Data Management Console using the Synonym Editor. Starting with a new synonym, each table is added using *Insert* > *Segment via Metadata Import* from the remote server. The JOIN_WHERE clause is automatically added to indicate the tables are joined on the JOBCLASS.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Executing Stored Procedures

In addition to making data available using metadata, a server can run commands that execute on the remote server. A typical command executes a procedure residing on the remote server, but CREATE and DROP commands can also be executed.

While the original configuration of remote servers will add explicit ENGINE EDA SET CONNECTION_ATTRIBUTES *node* lines to the selected profile, the sever also looks at the its communication file first to form the list of available target nodes. The result (unlike other adapter engines) is that the default remote node is not typically the first ENGINE EDA SET CONNECTION_ATTRIBUTES *node* line encountered in the profile.

There are two command options for setting which connection is used, as well as commands for testing connection status and passing alternate ID/password information. Additionally, the amper variables &EDASERVER and &EDAUSER will be populated with the current connection target node and ID.

Syntax: How to Set the Connection for the Next Request

Issue the following command:

```
SQL EDA SET SERVER node
```

where:

node

Is a valid configured node.

Syntax: How to Set the Connection for the Remainder of the Session

Issue the following command:

```
SQL EDA SET DEFAULT_CONNECTION node
```

where:

node

Is a valid configured node.

Alternately, you can use the following syntax:

```
REMOTE DEST = node
```

Syntax: How to Test Connection Status

Issue the following command:

```
SQL EDA PING node
```

where:

node

Is a valid configured node.

A connection that is not in a "ready" state, will display an appropriate FOC error message and set &RETCODE and &FOCERRNUM which can in turn be used in Dialogue Manager syntax to direct the execution as needed for the application (that is, -GOTO EXIT or use an alternate node).

Syntax: How to Use Alternate ID and Password Combinations With SET CONNECTION_ATTRIBUTES

Issue the following command:

```
SQL EDA SET CONNECTION_ATTRIBUTES node/id,password
```

where:

node

Is a valid configured node.

id

Is a valid ID on the remote server.

password

Is a valid password for the ID.

It is important to remember that setting a connection does not make it the default connection so, typically, this command needs to be used with a SET SERVER or SET DEFAULT_CONNECTION command.

Syntax: How to Use Alternate ID and Password Combinations With REMOTE USER and REMOTE PASS

Enter the following commands, where the ID and password apply to the current default remote connection.

```
REMOTE USER = id
REMOTE PASS = password
```

where:

id

Is a valid user ID on the remote server.

password

Is a valid password for the ID.

Syntax: How to Check Available Servers

```
SQL EDA ? SERVERS
```

Syntax: How to Execute a Stored Procedure Using Remote Execution

You can execute stored procedures by issuing the command:

```
SQL EDA EX rpcname parm1, parm2, ...
END
```

where:

rpcname

Is a procedure on the server.

parm1, parm2,...

Are character strings sent to the server (they are the same as parameters you can pass on the execution line of a FOCEXEC).

Syntax: How to Execute Locally Stored Commands on a Remote Connection

You can execute locally stored commands on a remote connection using the following syntax:

```
-REMOTE BEGIN
command1
command2
command3
...
-REMOTE END
```

where:

command1, command2, command3 ...

Are any number of commands or lines within a procedure (for example, the lines within a TABLE FILE ... END request).

The default remote connection is the target node where the code executes. Use SET DEFAULT_CONNECTION or REMOTE DEST to change the execution destination.

Any amper variables within the commands are resolved locally before shipment to the remote connection for execution.

Direct Dialogue Manager dash commands (for example, -SET, -IF, -GOTO) and amper variables that are expected to be resolved on the remote machine are not allowed because they are resolved first by Dialogue Manager on the local machine. You can use Dialogue Manager commands and variables indirectly by delaying resolution using a technique where the commands are hidden in local amper variables as in this example:

```
-SET &GOWHERE = 'STEP1' ;
-SET &DASHGOTO = '-GOTO' ;
-SET &DASHSTEP1 = '-STEP' ;
-SET &DASHSTEP2 = '-STEP2' ;
-SET &DASHSTART = '-START' ;
-SET &DASHEND = '-END' ;
-REMOTE BEGIN
&DASHSTART
&DASHGOTO &GOWHERE
&DASHSTEP1
...
&DASHGOTO END
&DASHSTEP2
&DASHEND
-REMOTE END
```

Syntax: How to Query Adapter Settings

To view the current adapter parameter settings, issue the command:

```
SQL EDA ?
```

The output is:

```
(FOC1450) CURRENT EDA INTERFACE SETTINGS ARE :
(FOC1446) DEFAULT DBSPACE IS                  -  : IBIEDA
(FOC1449) CURRENT SQLID IS                    -  : USER1
(FOC1444) AUTOCLOSE OPTION IS                 -  : ON FIN
(FOC1496) AUTODISCONNECT OPTION IS            -  : ON FIN
(FOC1499) AUTOCOMMIT OPTION IS                -  : ON COMMAND
(FOC1441) WRITE FUNCTIONALITY IS              -  : OFF
(FOC1445) OPTIMIZATION OPTION IS              -  : ON
(FOC1484) SQL ERROR MESSAGE TYPE IS           -  : DBMS
(FOC1552) INTERFACE DEFAULT DATE TYPE         -  : NEW
```

Chapter 86

Using the Adapter for REST

REST is an acronym for REpresentational State Transfer. REST-style architectures consist of clients and servers. Clients initiate requests to servers and servers process requests and return appropriate responses. REST was initially described in the context of HTTP. It utilizes well-known, well-defined methods like GET, POST, PUT, and DELETE.

A RESTful Web Service is a web service implemented using HTTP and the principles of REST.

The Adapter for REST supports RESTful Web Services, where the response is either XML or JSON (JavaScript Object Notation). For more information, see *Using the Adapter for JSON* on page 1245 and *Using the Adapter for XML* on page 2657.

In this chapter:

Configuring the Adapter for REST

Configuring the adapter consists of specifying connection information and if necessary, authentication information for at least one connection.

Declaring Connection Attributes

In order to access the RESTful Web Service provider hosting the target web service, the adapter requires connection information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).

You can declare connections to more than one web services provider by adding multiple connections, which generates multiple SET CONNECTION_ATTRIBUTES commands.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for the Adapter for REST

The Adapter for REST is under the Procedures group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Base Url

Is the part of the URL that is common for calling all the functionality of a particular RESTful Web Service. Addresses can begin with http:// or https://. For example: http://api.geonames.org.

Security

There are four methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.

- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.
- **OAuth.** The adapter connects to the database using the OAuth 2.0 authentication protocol for granting access to REST APIs.

KERBEROS SPN

Is the name by which a client uniquely identifies an instance of a service. For example: mydaemon/foo:4761.

KERBEROS SPN appears when Trusted Security is selected.

OAuth Grant Type

There are three OAuth Grant Types that can be configured:

- **Authorization Code.** Allows you to obtain a long-lived Access Token that can be renewed with a Refresh Token using an Authorization Code returned from an Authorization request.
- **Password.** Is used when the OAuth implementation only requires a User ID and Password.
- **Client Credentials.** Is used when the OAuth implementation only requires a Client ID and Client Secret

This field appears when OAuth security is selected.

Service Provider

OAuth configuration for certain Service Providers. This field appears when OAuth security is selected with the Authorization Code grant type.

Service URL

The URL to the Web Service when not configured as a REST or OData adapter connection.

User

Is the primary authorization ID by which you are known to the REST Service. This field appears when Explicit security or OAuth security with the Password Grant Type is selected.

Password

Is the password associated with the primary authorization ID. This field appears when Explicit security or OAuth security with the Password Grant Type is selected.

Client ID

Is the client ID defined in the REST application. This field appears when OAuth security is selected.

Client Secret

Is the Client Secret defined in the REST application. This field appears when OAuth security is selected.

Authorization URL

URL used for OAuth Authorization to a specific application.

For example, the Authorization URL for the Google set of APIs is https://accounts.google.com/o/oauth2/auth.

Token URL

Is the URL used for obtaining an Access Token to a specific application. This field appears when OAuth security is selected.

For example, the Token URL for the Google set of APIs is https://accounts.google.com/o/oauth2/token

Additional Authentication Parameters

This field appears when OAuth security with the Authorization Code Grant Type is selected. Enter any additional parameters required for authentication.

Additional Token Parameters

This field appears when OAuth security with the Authorization Code Grant Type is selected. Enter any additional parameters required for token retrieval.

Access Token

Is the value that identifies the user on whose behalf your REST application is acting. This field appears when OAuth security with the Authorization Code Grant Type is selected. Click *Get Access Token* to obtain this token.

In order for *Get Access Token* to complete successfully, the host name used to bring up the Reporting Server Web Console must match the host name set up for the Redirect URI in the REST application.

Refresh Token

Is the Refresh Token returned from the OAuth Token request. This field appears when OAuth security with the Authorization Code Grant Type is selected. The token is used for obtaining a new Access Token at the time a report is run accessing the REST Service for a specific application.

Chained Authentication

When checked, the authentication Web Service request is run in order to pass its response containing a cookie or a token to subsequent Web Service requests.

Authentication Synonym

This option is used only for chained authentication. When an application is selected, a list of Synonyms that have SUFFIX=REST is displayed. The selected Synonym will be used to perform Web Service authentication and return the security context, for example, baseapp/logon.mas.

For details about this process, see *Configure the Adapter for Chained Execution*.

HTTP Authorization Value

Is the name of the token value that is passed as part of the authorization within the HTTP header.

Authorization Row Key

Is the field name and value used to select the token from the response of the authentication request that is then passed as part of the authorization within the HTTP header, for example, INSTALLATIONID="1".

Select Authentication Row

When checked, displays the values for each of the fields from the Authentication Synonym for the purpose of populating the Authorization Row Key value. When a value is clicked, select *Show RowKey*.

The Authorization Row Key is populated with the FIELDNAME and its corresponding value, for example, INSTALLATIONID="1".

Advanced HTTP connection options

Add Custom Headers

Opens a text box for adding Custom Headers.

Custom Headers

Custom Header values should be enclosed in double quotation marks, and the individual custom headers should be delimited by semi-colons (;). For example:

```
Content-Type="CDF";PARAM1="ABC"
```

Defining custom headers in the connection string makes the custom headers apply to the entire REST service. Custom headers can also be defined at the metadata level. Defining the custom headers at the metadata level makes the custom headers apply to the specific REST call, and the custom header values can be changed at report time using WHERE or IF statements.

PROXY Server IP Address

Is the IP address of the proxy server, which intercepts requests and forwards them to the actual server.

PROXY Port

Is the port number on which the proxy server listens. The default port number is 80.

PROXY HTTPS Relative Path

When checked, the REST request will send the relative path rather than the absolute path when a proxy server is configured.

SSL Certificate

Is the location of locally-stored, user-provided server x.509 certificates for SSL authentication. The certificate file is used to authenticate the server to which the adapter is connecting.

SSL Mutual Authentication

When checked, Mutual Authentication is enabled.

SSL Certificate type

Is the certificate type. The options are:

- *trusted.* The trusted certificate file (trustedcertfile) points to a file of CA certificates in PEM format, as illustrated below:

  ```
  -----BEGIN CERTIFICATE-----
  ... (CA certificate in base64 encoding) ...
  -----END CERTIFICATE-----
  ```

  A trusted certificate file can contain several CA certificates. You can add text before, between, and after certificates. For example, to provide descriptions of the certificates.

- *non-trusted.* This option adds Key file, Pass phrase, and Label to the configuration pane.

  SSL Certificate key file

  Is the private key used for creating the client X.509 certificate in PEM format. This option is used together with a certificate for a non-trusted connection.

  SSL Certificate pass phrase

  Is the password used to unlock the key file. The value is needed only if the key file is encrypted.

  SSL Certificate label

  Identifies a certificate in the file if the file contains more than one certificate.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Security for RESTful Web Services

Chained execution of an authentication operation is used when the web server returns a response containing a cookie or a token to be used in the subsequent processing operation. Chained execution is defined in the connection string associated with the requested operation. When a cookie is used, both connection strings (authentication and execution) must contain the same Base URL.

The Adapter for RESTful Web Services supports the following security facilities:

- Cookies, containing user credentials, which are in effect for the length of a TSCOM agent session (that is, between user connect and disconnect). The scope of the cookie is the single TSCOM agent. The received cookie is sent back to the sender IP, along with subsequent REST requests.
- Identification token values, which are returned by an authentication operation and are acceptable to associated execution operations.

Chained authentication supports the Explicit and Password Passthru (with PING capabilities) security models.

To use chained authentication, you must declare the connection for authentication, then create a Master File that contains the authentication specifications (using an authentication string), and store it in an application folder. You can then use the authentication metadata to create one or more associated execution connection strings.

Note: When the connection string contains user ID and password information, and chained authentication is not defined, basic HTTP authentication is used. That is, the user ID and password are encrypted using the x64 algorithm, and it is then used to establish the HTTP connection.

Procedure: How to Configure the Adapter for Chained Execution

To configure the adapter for chained authentication:

1. Create a connection string using a BASE URL needed for the REST call to authenticate. This connection is needed to specify the part of the REST URL that would prefix the URL Extension to the authentication operation.

   In the Connection Parameters pane of the Web Console or the Data Management Console, enter the BASE URL, and click *Configure*.

   Keep in mind that if a cookie (rather than a token) is to be used in the subsequent processing operation, you must specify the same Base URL for this authentication connection string and for the execution connection string that you specify in step 3.

2. Create synonyms for relevant authentication operations.

   Create the synonym from the Web Console or the Data Management Console, then open the Synonym Editor to edit the synonym using the following guidelines.

   Authentication synonyms must include the following input fields:

   USERNAME

   The user ID value is taken from the connection string defining the operation to be executed.

   PASSWORD

   The password value is taken from the connection string defining the operation to be executed.

   Authentication synonyms may also include fields with the following access properties. The following fields must belong to segments describing the REST request:

   ACCESS_PROPERTY=NEED_VALUE

   Defines fields that provide additional parameters for an authentication operation. You can supply default values in XDEFAULT parameters. Values provided in the associated connection string overwrite the default values.

   There may be more than one such field.

The following fields must belong to segments describing the REST response:

ACCESS_PROPERTY=AUTHRESP

Defines fields that describe the result of an authentication operation. Correct response values must be provided in the ACCEPT attribute (using the OR predicate if more than one value is acceptable). There may be more than one such field. The operation is considered invalid if at least one of the fields contains a non-acceptable value.

ACCESS_PROPERTY=AUTHTOKEN

Defines a field that contains a response token to be passed as an input value to the operation to be executed. There can be only one such field. If none is defined, the authentication operation is expected to return a cookie.

3. Create a connection string using a BASE URL that is used to call the execution request and a previously created authentication operation synonym.

   In the Connection Parameters pane of the Web Console or the Data Management Console, enter the BASE URL, then click the *Chained Authentication* checkbox.

   Click the ellipse next to the Authentication Synonym text box, and select the application and Master File on the server that describes the associated authentication operation.

   Click OK.

4. In order to create synonyms for relevant execution operations, this must be done within a procedure. The reason is that Create Synonym runs a REST request to determine the response returned. The request must be authenticated properly to generate a valid response for use in creating the metadata.

For example:

```
-*Authentication Request
TABLE FILE LOGON
PRINT
   ENTRY.VALUE AS 'CSRF Token'
   RETURNCODE
   RETURNDESC
IF USERNAME EQ 'admin'
IF PASSWORD EQ 'admin'
IF ENTRY.KEY EQ 'IBI_CSRF_Token_Value'
ON TABLE HOLD AS CSRFTOKEN
END
-RUN
-READ CSRFTOKEN &&CSRFTOKEN.A100
-*Create Synonym which passes Token
-*Since the Authentication request and Create Synonym run in the same
Agent, Cookies are automatically passed
CREATE SYNONYM restful_web_services/folderlist AT ibfs FOR rs PARMS
 'HTTPMETHOD=GET PARAMSURL="IBIRS_action=get&|IBIRS_service=ibfs&|
IBIWF_SES_AUTH_TOKEN=&&CSRFTOKEN"'
    CHECKNAMES UNIQUENAMES DBMS REST DROP
-RUN
```

5. Ensure that the operation synonyms have an input field that describes the authenticating token (if needed) using ACCESS_PROPERTY=AUTHTOKEN.

   From the Web Console or the Data Management Console, open the Synonym Editor to edit the synonym as described.

   For related information, see *Connection Attributes for the Adapter for REST* and *Creating Synonyms* on page 2028.

Example: Configuring Chained Authentication

Connection string:

```
ENGINE REST SET CONNECTION_ATTRIBUTES ibfs/,:'http://localhost:8080/
ibi_apps/rs auth:rest/logon'
```

Master File fragment describing an authentication operation:

```
FILENAME=GETTOKEN, SUFFIX=REST , $
  SEGMENT=GETTOKEN, SEGTYPE=S0, $
  GROUP=HEADER, ALIAS=Header, ELEMENTS=2, $
  FIELDNAME=USERNAME, ALIAS=username, USAGE=A30, ACTUAL=A30,
    ACCESS_PROPERTY=(NEED_VALUE), $
  FIELDNAME=PASSWORD, ALIAS=password, USAGE=A30, ACTUAL=A30,
    ACCESS_PROPERTY=(NEED_VALUE), $
  FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
    ACCESS_PROPERTY=(INTERNAL),$
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=GETTOKEN,
    POSITION=__RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=getToken0Out, USAGE=A1, ACTUAL=A1,
    ACCESS_PROPERTY=(INTERNAL), $
  FIELDNAME=RESULT, ALIAS=Result, USAGE=A120, ACTUAL=A120,
    ACCESS_PROPERTY=(AUTHTOKEN),
  REFERENCE=RESPONSE, PROPERTY=ELEMENT, $
```

Master File fragment describing a data retrieval operation:

```
FILENAME=FINDADDRESS, SUFFIX=REST , $
SEGMENT=FINDADDRESS, SEGTYPE=S0, $
GROUP=HEADER, ALIAS=Header, ELEMENTS=8, $
 FIELDNAME=HOUSENUMBER, ALIAS=houseNumber, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='2815', $
 FIELDNAME=STREET, ALIAS=street, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='PRAIRIE AVE.', $
 FIELDNAME=INTERSECTION, ALIAS=intersection, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
 FIELDNAME=CITY, ALIAS=city, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='MIAMI BEACH', $
 FIELDNAME=STATE_PROV, ALIAS=state_prov, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='FL', $
 FIELDNAME=ZONE, ALIAS=zone, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='33140', $
 FIELDNAME=COUNTRY, ALIAS=country, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='US', $
 FIELDNAME=TOKEN, ALIAS=token, USAGE=A120, ACTUAL=A120,
   ACCESS_PROPERTY=(NEED_VALUE, AUTHTOKEN), $
 FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
   ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=FINDADDRESS,
   POSITION=__RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=findAddressResponse, USAGE=A1, ACTUAL=A1,
   ACCESS_PROPERTY=(INTERNAL), $
```

Syntax:

How to Pass Additional Parameters to an Authentication Operation

You can pass an extra set of parameters to the authentication operation using XDEFAULT parameters to set the default values in the appropriate Master File.

Tip: To complete this task, use the Synonym Editor in the Web Console or the Data Management Console.

A sample of the syntax is in the following example.

Example: **Passing Additional Parameters to an Authentication Operation**

```
ENGINE REST SET CONNECTION_ATTRIBUTES belgR/:'https://secure.securex.be/
HRAWebservice/webservices AUTH:belgium/
directauthenticate DBID="HRADemo01" LANGID="1"'
```

Master File fragment:

```
FILENAME=DIRECTAUTHENTICATE, SUFFIX=REST , $
SEGMENT=DIRECTAUTHENTICATE, SEGTYPE=S0, $
GROUP=HEADER, ALIAS=Header, ELEMENTS=4, $
FIELDNAME=USERNAME, ALIAS=usrname, USAGE=A30, ACTUAL=A30,
   ACCESS_PROPERTY=(NEED_VALUE), $
FIELDNAME=PASSWORD, ALIAS=pwd, USAGE=A30, ACTUAL=A30,
   ACCESS_PROPERTY=(NEED_VALUE), $
FIELDNAME=DBID, ALIAS=dbId, USAGE=A30, ACTUAL=A30,
ACCESS_PROPERTY=(NEED_VALUE), $
   XDEFAULT='Test', $
FIELDNAME=LANGID, ALIAS=langId, USAGE=A30, ACTUAL=A30,
ACCESS_PROPERTY=(NEED_VALUE), $
   XDEFAULT='E', $
FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
   ACCESS_PROPERTY=(INTERNAL), $
SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=DIRECTAUTHENTICATE,
   POSITION=__RESPONSE, $
FIELDNAME=RESPONSE, ALIAS=DirectAuthenticateResponse, USAGE=A1,
   ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL), $
FIELDNAME=DIRECTAUTHENTICATERESULT, ALIAS=DirectAuthenticateResult,
  USAGE=A5, ACTUAL=A5, ACCESS_PROPERTY=(AUTHRESP), ACCEPT='true',
  REFERENCE=RESPONSE, PROPERTY=ELEMENT, $
```

Managing RESTful Web Services Metadata

When the server accesses a RESTful Web Services provider, it needs to know how to pass parameters and accept responses for each REST request. In most RESTful Web Services, the Web Services provider documentation is used in determining the parameters to pass and the expected response. Therefore, the Create Synonym process sends a REST request including parameters to determine the response from the Web Service.

Creating Synonyms

A synonym defines a unique logical name (also known as an alias) for each Web Services operation. Synonyms are useful because:

- They insulate client applications from changes to the location and identity of a request. You can move or rename a request without modifying the client applications that use it. You need make only one change, redefining the request synonym on the server.

- They provide support for the extended metadata features of the server, such as virtual fields and security mechanisms.

Creating a synonym generates a Master File and an Access File. These are metadata files that describe the Web Services request to the server.

Each synonym you create represents a single operation.

Note: Each synonym describes the Web Services operations of a particular provider. If the operation parameters are changed, the synonym must be recreated.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for RESTful Web Services

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select REST Operation

Select one of the following REST operations:

- **Get.** GET operation.
- **Post.** POST operation.

Service URL Extension

Is an extension to the Base URL used to perform specific functionality from the RESTful Web Service.

The Base URL is concatenated with the Service URL Extension at run time to make the REST request. This parameter *must* have a value. For example:

```
postalCodeSearch
```

Parameterize

When checked, parameterizes the Service URL Extension.

Additional fields will be added to the Master file for each part of the path of the Service URL extension. The field names created will start with the character ID and have a numeric suffix equal to the part of the path that is being parameterized.

For example, if the Service URL extension is a/b/c, the following fields will be added to the Master file:

```
FIELDNAME=ID1, ALIAS=id, USAGE=A1, ACTUAL=A1,
  ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='a', $
FIELDNAME=ID2, ALIAS=id, USAGE=A1, ACTUAL=A1,
  ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='b', $
FIELDNAME=ID3, ALIAS=id, USAGE=A1, ACTUAL=A1,
  ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='c', $
```

The OBJECT attribute in the Access file will be a path containing the fields added in the Master file enclosed with an ampersand (&) character. For example, if the Service URL extension is a/b/c, the following OBJECT attribute will be added in the Access file:

```
OBJECT=&ID1&/&ID2&/&ID3&
```

Service URL Parameters

Are the parameter name and value pairs that will be used in making a REST request in order to create metadata from the response.

The input parameter names will also be included in the metadata.

It is important to note that with certain RESTful Web Services, different parameters and/or parameter values could return a different structured response. Therefore, different synonyms should be created for each differently structured response. For example:

```
postalcode=9011&maxRows=10&username=demo
```

Provide document sample

This parameter becomes visible only if Post is selected as the operation.

It is used in conjunction with the Service URL Parameters field by passing the Body portion of the REST request to create the metadata. The body can be made up of any combination of name-value pairs, XML, and JSON. For example:

```
IBIRS_action=put&object=<object _jt="IBFSUserObject" description="Rest
Userid" email="restid@informationbuilders.com" password="rest"
type="User"> <status _jt="IBSSUserStatus" name="ACTIVE"/> </
object>&IBIRS_replace=true
```

Custom Headers

Custom Header values should be enclosed in double quotation marks, and the individual custom headers should be delimited by semi-colons (;). For example:

```
Content-Type="CDF";PARAM1="ABC"
```

Defining custom headers in the connection string makes the custom headers apply to the entire REST service. Custom headers can also be defined at the metadata level. Defining the custom headers at the metadata level makes the custom headers apply to the specific REST call, and the custom header values can be changed at report time using WHERE or IF phrases.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Synonym Name

Enter a name to assign to the synonym.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Master File Attributes

Attribute	Description
PROPERTY	Indicates whether the field corresponds to an XML/JSON attribute or an XML/JSON element.
REFERENCE	Identifies the parent element of the field in the XML/JSON hierarchy. The format is *segmentname.fieldname* where: *segmentname* Is the name of the Master File segment in which the field resides. *fieldname* Is the name of the field that corresponds to the parent element.

Reference: Access File Attributes

Attribute	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection*
OBJECT	Service URL Extension as defined during the Create Synonym process.
HEADER	The GROUP name in the Master File that defines the parameters for the REST GET request.
SERVICETYPE	The Service Type which will always be REST.
HTTPMETHOD	The HTTP method used in the REST request, for example, GET or POST.

Attribute	Description
RESTRESPONSE	The type of response returned from the REST request. Valid values are XML or JSON.
HTTPBODY	If HTTPMETHOD=POST, setting HTTPBODY=ENCODE will encode the body of the request.
TIMEOUT	Web Services timeout in seconds

Reference: **Sample REST Synonym**

The following synonym was generated using the following connection and synonym creation parameters:

`Base URL`

`http//api.geonames.org`

The base URL is entered on the Add Connection page.

The remaining entries are on the Create Synonym page.

`Select REST Operation`

Select *Get*.

`Service URL Parameters`

Enter the following parameters:

`postalcode=10121&country=US&maxRows=10&username=demo`

`Synonym Name`

Enter *postalCodeSearch*.

In addition, check the *Validate* and *Make Unique* check boxes, then click *Create Synonym*.

postalCodeSearch Master File

```
FILENAME=M6ILO, SUFFIX=REST    , $
  SEGMENT=M6ILO, SEGTYPE=S0, $
   GROUP=HEADER, ALIAS=Header, ELEMENTS=4, $
    FIELDNAME=POSTALCODE, ALIAS=postalcode, USAGE=A30, ACTUAL=A30,
ACCESS_PROPERTY=(NEED_VALUE),
      XDEFAULT='10121', $
    FIELDNAME=COUNTRY, ALIAS=country, USAGE=A30, ACTUAL=A30,
ACCESS_PROPERTY=(NEED_VALUE),
      XDEFAULT='US', $
    FIELDNAME=MAXROWS, ALIAS=maxRows, USAGE=A30, ACTUAL=A30,
ACCESS_PROPERTY=(NEED_VALUE),
      XDEFAULT='10', $
    FIELDNAME=USERNAME, ALIAS=username, USAGE=A30, ACTUAL=A30,
ACCESS_PROPERTY=(NEED_VALUE),
      XDEFAULT='demo', $
    FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML     , PARENT=M6ILO,
POSITION=__RESPONSE, $
    FIELDNAME=GEONAMES, ALIAS=geonames, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL),
      PROPERTY=ELEMENT,  $
    FIELDNAME=TOTALRESULTSCOUNT, ALIAS=totalResultsCount, USAGE=P33,
ACTUAL=A33,
      REFERENCE=GEONAMES, PROPERTY=ELEMENT,  $
    FIELDNAME=CODE, ALIAS=code, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GEONAMES, PROPERTY=ELEMENT,  $
    FIELDNAME=POSTALCODE1, ALIAS=postalcode, USAGE=P33, ACTUAL=A33,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=NAME, ALIAS=name, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=COUNTRYCODE, ALIAS=countryCode, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=LAT, ALIAS=lat, USAGE=P20.3, ACTUAL=A20,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=LNG, ALIAS=lng, USAGE=P20.3, ACTUAL=A20,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $

    FIELDNAME=ADMINCODE1, ALIAS=adminCode1, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=ADMINNAME1, ALIAS=adminName1, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=ADMINCODE2, ALIAS=adminCode2, USAGE=P33, ACTUAL=A33,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=ADMINNAME2, ALIAS=adminName2, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=ADMINCODE3, ALIAS=adminCode3, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
    FIELDNAME=ADMINNAME3, ALIAS=adminName3, USAGE=A55, ACTUAL=A55,
      REFERENCE=CODE, PROPERTY=ELEMENT,  $
```

postalCodeSearch Access File

```
SEGNAME=M6ILO,
  CONNECTION=geonames,
  OBJECT=postalCodeSearch,
  HEADER=HEADER,
  SERVICETYPE=REST,
  HTTPMETHOD=GET,
  RESTRESPONSE=XML, $
```

Chapter 87

Using the Adapter for RMS

The Adapter for RMS allows applications to access RMS data sources. The adapter converts application requests into native RMS statements and returns answer sets to the requesting application. If the metadata is configured for read/write capabilities, it can insert data from an application into the data source.

You can also access compressed data sets with this adapter. However, the actual compression is done using the ZCOMP Exit. For details, see *Data Set Compression Exit: ZCOMP* on page 2719.

In this chapter:

Preparing the RMS Environment

RMS is a standard part of OpenVMS. Therefore, the standard release level for the Version of OpenVMS being used is the supported release level.

You can create metadata based on either CDD Repositories or COBOL File Descriptions. If you use CDDs, a proper RDB environment must be available. If RDB standard (of any version level) and CDDs are installed and available globally to all users, then no additional steps are required for CDD use.

Configuring the Adapter for RMS

You can configure the Adapter for RMS from the Web Console or the Data Management Console.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Managing RMS Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the RMS data types.

You can create metadata:

- By using the Create Synonym facility on the Web Console or the Data Management Console.

- By manually creating a Master File and Access File with a system editor.

Both methods describe data files to the server for OpenVMS with a Master and an Access File.

The Master File describes the type of data file you are using, the structure, and the fields it contains. The Access File provides a number of attribute parameters, including SHARED ACCESS to support READ/WRITE capabilities and an RMSFILE attribute, which offers one method for associating the data source with the Master File. For details about Access File uses, see *When to Use an Access File With an RMS Data Source* on page 2080.

Creating Synonyms

Synonyms define unique names (or aliases) for each RMS table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms. Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for RMS

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select potential candidates for synonyms:

Synonym Candidates

You can base your synonym on either CDD definitions or COBOL definitions. Choose one of these options from the drop-down list.

Next, you must specify the locations of either the CDD or COBOL definitions. Options vary depending on the selection you make here.

For related information about COBOL File Descriptions, see *Translating COBOL File Descriptions* on page 2699.

For CDD definitions:

Location of CDDs

If your repository is in a non-standard location (that is, one which is not declared by CDD $DEFAULT), you can supply the CDD location here.

If CDD$DEFAULT is declared, its current value appears in the input box, where you may leave it as is or change it.

CDD name

Type a filter for retrieving a partial list of CDD references (for example, those starting with V%).

You must supply filter values in *uppercase* characters, followed by a % sign. OpenVMS wildcards are not permitted.

For COBOL definitions:

Directory Path

Specify a full path directory name where the COBOL definitions are located. (Wild card characters are not permitted.)

For information about specific layout and syntax requirements, see *COBOL FD Syntax Requirements* on page 2717.

File Name/File Extension

If you wish to limit retrieval, you can type a file name and/or file extension:

- In the File name box, type a full name or a partial name with a wildcard symbol %. A full name returns just that entry. A name with a wildcard symbol may return many entries.
- In the File extension box, type an extension with or without the wildcard symbol %.

You must supply filter values in *uppercase* characters.

Location of RMS data files:

Directory path

Specify a full path directory name where RMS is located. This may be a logical that points to the full directory path or a directory path that uses a logical. OpenVMS wildcards are not allowed.

File Name

Type a filter for retrieving a partial list of RMS file names in the directory (for example, those starting with V%).

You must supply filter values in *lowercase* characters, followed by a % sign. OpenVMS wildcards are not allowed.

File Extension

Type a filter for retrieving a partial list of RMS files based on the file extensions within the directory.

The normal default for RMS files, dat%, is pre-populated in the input box and must *not* be changed if you wish to search for .dat files.

You can change the .dat value by specifying a filter to access files that do not use the RMS normal default (that is, those starting with rms%). You must supply filter values in *lowercase* characters, followed by a % sign. OpenVMS wildcards are not allowed.

Get Candidates button

Click this button to obtain the selection list of candidates for which synonyms can be created.

Select actual candidates for synonyms:

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Tip: While you can change individual names in the list, entering a prefix or suffix enables you to make global changes that are particularly useful when many synonyms are being created.

An alternate method for dealing with duplicate names is to store them in separate application directories and change the application path as needed for a particular table.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Set COBOL FD translation options

For a COBOL FD, optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

For CDD definitions:

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select CDD records

To select all CDD records in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific records, select the corresponding check boxes.

Associate records to physical files

In the list of CDD objects and data files, use the pull-down menu on the right to associate the CDD record with the physical data file.

For COBOL definitions:

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select COBOL records

To select all COBOL records in the list, select the check box to the left of the *Default Synonym Name* column heading.

To select specific records, select the corresponding check boxes.

Associate records to physical files

In the list of COBOL objects and data files, use the pull-down menu on the right to associate the COBOL record with the physical data file.

Example: Creating a Synonym Using a COBOL File Description

To generate the following synonym from the Web Console or the Data Management Console Create Synonym panes:

1. On the first Create Synonym pane, select *COBOL FD* from the Synonym Candidate drop down list.
2. To specify the location of the COBOL FD, enter *mydata:[RMS]* as the Directory Path.
3. Leave the File Name field blank and accept the default extension, cbl, to filter the list of COBOL FDs.
4. Next, specify the location of the RMS data files. Enter *mydata:[RMS]* as the Directory Path.
5. Leave the File Name field blank and accept the default extension, dat.
6. Click the *Get Candidates* button.
7. The list of candidates is displayed on the second Create Synonym pane. Click the check box for *rmscar.cbl* in the first column.
8. To associate the selected candidate with a particular COBOL data file, choose *rmscar.dat* from the pull-down menu in the RMS data file column.

   Tip: If you wish, you can choose a different data file at a later time using a FILEDEF, as described in *Associating an RMS Data Source to a Master File* on page 2078.
9. For this example, the default translation options are suitable, so leave the Customization options box unchecked.
10. Click the *Create Synonym* button.

    The synonym is created and added under the specified application directory (baseapp is the default).

    A status window displays the message: *All Synonyms Created Successfully*
11. From the message window, click *Applications* on the menu bar.
12. Open the *baseapp* application folder in the navigation pane and click the synonym *rmscar*.
13. Choose *Edit as Text* from the menu to view the generated Master File.

    Generated Master File:

```
FILENAME=RMSCAR, SUFFIX=RMS     ,
 DATASET=TSCQDATA:[TSCQDATA.RMS]rmscar.dat, $
  SEGMENT=ROOT, SEGTYPE=S0, $
   GROUP=ROOT, ALIAS=E2, USAGE=A62, ACTUAL=A62, $
   GROUP=PRIMARYKEY, ALIAS=KEY, USAGE=A50, ACTUAL=A50, $
    FIELDNAME=COUNTRY, ALIAS=E3, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CAR, ALIAS=E4, USAGE=A16, ACTUAL=A16, $
    FIELDNAME=MODEL, ALIAS=E5, USAGE=A24, ACTUAL=A24, $
    FIELDNAME=BODYTYPE, ALIAS=E6, USAGE=A12, ACTUAL=A12, $
    FIELDNAME=SEATS, ALIAS=E7, USAGE=I8, ACTUAL=I4, $
    FIELDNAME=RETAIL_COST, ALIAS=E8, USAGE=I8, ACTUAL=I4, $
    FIELDNAME=DEALER_COST, ALIAS=E9, USAGE=I8, ACTUAL=I4, $
    FIELDNAME=SALES, ALIAS=E10, USAGE=I8, ACTUAL=I4, $
```

14. To see the Access File, choose *Edit Access File as Text*.

    Generated Access File:

```
ACCESS=SHARED, $
```

Example: Creating a Synonym Using CDD Record Definitions

To generate the following synonym from the Web Console or the Data Management Console Create Synonym panes:

1. On the first Create Synonym pane, select *CDD Definition* from the Synonym Candidate drop down list.
2. To specify the location of the CDDs, enter *mydata:[RMS.CDDPLUS]* as the Directory Path.
3. In the CDD Name field, enter R% to filter the list of CDD definitions. (Uppercase characters are required.)
4. Next, specify the location of the RMS data files. Enter *mydata:[RMS.CDDPLUS* as the Directory Path.
5. Click the *Get Candidates* button.
6. The list of candidates is displayed on the second Create Synonym pane. Click the check box for *rmscar* in the first column.
7. To associate the selected candidate with a particular CDD data file, choose *rmscar.dat* from the pull-down menu in the RMS data file column.

   Tip: If you wish, you can choose a different data file at a later time using a FILEDEF, as described in *Associating an RMS Data Source to a Master File* on page 2078.
8. Click the *Create Synonym* button.

   The synonym is created and added under the specified application directory (baseapp is the default).

A status window displays the message: *All Synonyms Created Successfully*

9. From the message window, click *Applications* on the menu bar.
10. Open the *baseapp* application folder in the navigation pane and click the synonym *rmscar*.
11. Choose *Edit as Text* from the menu to view the generated Master File.

Generated Master File:

```
FILENAME=RMSCAR, SUFFIX=RMS     ,
 DATASET=TSCQDATA:[TSCQDATA.RMS]rmscar.dat, $
  SEGMENT=ROOT, SEGTYPE=S0, $
   GROUP=PRIMARYKEY, ALIAS=KEY, USAGE=A50, ACTUAL=A50, $
    FIELDNAME=COUNTRY, ALIAS=COUNTRY, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CAR, ALIAS=CAR, USAGE=A16, ACTUAL=A16, $
    FIELDNAME=MODEL, ALIAS=MODEL, USAGE=A24, ACTUAL=A24, $
    FIELDNAME=SEATS, ALIAS=SEATS, USAGE=P7, ACTUAL=I4, $
    FIELDNAME=RETAIL_COST, ALIAS=RCOST, USAGE=P7, ACTUAL=I4, $
    FIELDNAME=DEALER_COST, ALIAS=DCOST, USAGE=P7, ACTUAL=I4, $
    FIELDNAME=SALES, ALIAS=SALES, USAGE=P7, ACTUAL=I4, $
```

12. To see the Access File, choose *Edit Access File as Text*.

Generated Access File:

```
ACCESS=SHARED, $
```

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Manually Describing RMS Files

These topics outline the procedures necessary for manually describing RMS files.

File Attributes

File attributes are FILENAME, SUFFIX, and, optionally, DATASET, which name the file and describe the file type and data source location. For example:

```
FILENAME=LIBRARY1, SUFFIX=RMS, DATASET=mydata.DAT ,$
```

FILENAME

The FILENAME attribute is optional. It is recommended that you include the file name for documentation purposes. You can specify any name for this attribute, but you should use the same name that you give the Master File.

The syntax is

```
FILE[NAME] = name
```

where:

name

Is any name of 1 to 8 characters.

SUFFIX

The SUFFIX attribute describes the type of data file it will read.

The syntax is:

```
SUFFIX = type
```

where:

type

Is a suffix listed in the following table:

Type of File	Suffix	Physical Access Method
Keyed (Indexed) or unkeyed RMS file	RMS	RMSFILE attribute in Access File or DATASET attribute in the Master File or FILEDEF

The description of data and relationships between fields within a file is the same for keyed (indexed) RMS files, FIX (fixed-format sequential) files, and COM (comma-delimited) files, except for the following differences:

- Keyed (indexed) RMS uses the keyed RMS File System to access data using information in the Access File declarations, where FIX and COM are fopen() accessed. Alternately you can use an RMSFILE= attribute in the Access File or override the DATASET= value with a FILEDEF. For more information, see *Associating an RMS Data Source to a Master File* on page 2078.
- Index related declarations do not apply to fixed-sequential or comma-delimited files.

DATASET

DATASET is an optional attribute that specifies the physical or app location of the data source. It is used at the file declaration level of the Master File. The syntax is

```
{DATASET|DATA}={'filename'|app/filename[.extension]}
```

where:

filename

Is the platform-dependent physical name of the data source. In a prior release, the physical name was required to be in single quotes. This is still valid syntax, but not required. A direct logical name such as MYQ2DATA: is also valid but must include the colon character, so it is distinct from an app-based name.

app/filename[*.extension*]

Is an app-based logical name of the data source. The default extension when no extension is supplied is .dat.

Segment Attributes

A Master File can be divided into segments, which are groups of fields that are related to one another. It is not always necessary to divide your file into segments.

Segment declarations identify and describe each segment in a file. They name the segments and indicate the relative positions in the file structure. In files with multiple record types, each record type will be described as a separate segment. Files with mixed singly- and multiply-occurring fields must also be described with separate segments defined for each type of occurrence. The attributes used to describe segments of different record types and multiply-occurring fields are described in *Describing Multiple Record Types* on page 2064 and in *Describing Embedded Repeating Data* on page 2069.

The following is an example of a segment declaration:

```
SEGNAME=BOOKINFO, PARENT=PUBINFO, SEGTYPE=S0,$
```

SEGNAME

Each segment declaration starts with the SEGNAME (or SEGMENT) attribute, which names the segment.

The syntax is

```
{SEGNAME|SEGMENT}= name
```

where:

name

Is a unique name of 1 to 8 characters.

PARENT

Files with more than one segment are defined as multi-segment structures.

Segments in a multi-segment structure have a parent/child relationship. Each segment, except the top or "root" segment, is the descendant of another segment called the "parent." The PARENT attribute is used to identify a segment's parent segment. If no PARENT attribute is specified in the Master File, the default parent segment is the immediately preceding segment, except for the top segment, which has no parent.

The syntax is

```
PARENT = name
```

where:

name

Specifies a SEGNAME in the file.

For example:

```
...PARENT=PUBINFO...
```

SEGTYPE

The SEGTYPE attribute for RMS files is specified as S0.

The required syntax is:

```
SEGTYPE=S0
```

Field Attributes

Field attributes describe the actual fields in each segment. Each field declaration consists of at least four attributes. They are:

```
FIELDNAME   ALIAS   USAGE   ACTUAL
```

For example:

```
FIELDNAME=PUBNO ,ALIAS=PN ,USAGE=A10 ,ACTUAL=A10 ,$
```

There are also other optional attributes that can be used, such as DESCRIPTION, TITLE, and ACCEPT. DESCRIPTION enables you to include a description of the field. TITLE is the default report column title other than the field name. ACCEPT assigns a list or range of acceptable values to a field. ACCEPT is also used for RECTYPE values. These optional attributes are used by FOCUS, WebFOCUS, and various other client products. For further information about optional attributes, see the appropriate manuals.

FIELDNAME

Field names are unique names of 1 to 66 characters with the exception of indexes, which are limited to12. The field name appears as a default column heading when you name the field in a report request. Field names may consist of any alphanumeric characters, but the first character must be a letter from A to Z. Field names may include embedded blanks, but it is not recommended, and you will need to enclose such a field name in single quotation marks (') if you reference the complete name (including the blank) in a request. Avoid using special characters (+ - $ * /() ' ; . , = and " > <), since they may cause confusion if the field is used in calculations.

The syntax is

```
FIELD[NAME] = name
```

where:

name

Meets the criteria described above.

ALIAS

Aliases are optional field names. Each field can have an alias to be used interchangeably with the field name. The length and format rules for field names apply to aliases. Aliases are not used as column titles.

The syntax is

```
ALIAS = alias
```

where:

alias

Is a name of 1 to 66 alphanumeric characters meeting the same criteria as for field names.

If you omit the alias, you must indicate its absence with either the following entry in the field declaration

```
ALIAS=,
```

or by holding its place with a comma delimiter (,):

```
FIELDNAME=PUBNO , ,USAGE=A10 ,ACTUAL=A10 ,$
```

USAGE

The USAGE attribute describes the way you want to use the field and display its values on reports. This attribute includes the data field type, display length, and any edit options that are to be applied when the field values are printed.

The syntax is

```
USAGE = usage
```

where:

usage

Describes the field in three parts: field type, field display length, and edit options.

The values that you specify for type and display length determine the number of print positions allocated for the field in any display or report. Edit options only affect printed or displayed fields; they are not active for extract files or other non-display retrievals.

The following table shows permissible USAGE field types and display lengths:

USAGE	Length	Description
A	1-9,095	Alphanumeric text
D	1-19	Decimal, double-precision numbers

USAGE	Length	Description
F	1-9	Decimal, single-precision numbers
I	1-11	Integer values (no decimal places)
P	1-17	Packed decimal numbers
YYMD	10	Displayed as YYMD
D,W,M,Q or Y	Date	Date display
H	Date-Time	Date-Time stamp data that uses formatting options to display raw data or formatted time data, date data, or date-time data. See The *Describing Data With WebFOCUS Language* manual for a full list of options.

The following table shows edit options that may be applied to various A, D, F, I, or P usages:

Edit Option	Meaning	Effect
%	Percent sign	Percent sign Displays a percent sign along with numeric data. Does not calculate the percent.
B	Bracket negative	Encloses negative numbers in parentheses.
C	Comma suppress	Suppresses the display of commas. Used with numeric format options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision).
C	Comma edit	Inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use.
DMY	Day-Month-Year	Displays alphanumeric or integer data as a date in the form day/month/year.
E	Scientific notation	Scientific notation Displays only significant digits.

Edit Option	Meaning	Effect
L	Leading zeroes	Adds leading zeroes.
M	Floating $ (for US code page)	Places a floating dollar sign $ to the left of the highest significant digit. **Note:** The currency symbol displayed depends on the code page used.
MDY	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
N	Fixed $ (for US code page)	Places a dollar sign $ to the left of the field. The symbol displays only on the first detail line of each page. **Note:** The currency symbol displayed depends on the code page used.
R	Credit (CR) negative	Places CR after negative numbers.
S	Zero suppress	Zero suppress If the data value is zero, prints a blank in its place.

Note:

- Edit options can be specified in any order.
- The total length of the USAGE specification, including all edit options, may not exceed eight characters.
- Options M and N (floating and fixed dollar sign) both automatically imply option C (comma edit).
- Format type D (double-precision decimal number) implies option C.

The following table shows the various USAGE formats as they would be specified in a Master File. The USAGE formats contain type and length information as well as various edit options. The Value column shows the actual value as it would be read from the external file, and the Display column shows how the number would be displayed on a report.

USAGE	Value	Display
I7C	47693	47,693
D10.2S	0.00	
P8.0CR	-4719	4,719 CR
F8.2MC	28148.00	$28,148.00
I5B	-341	(341)
D7M	8741	$8,741
D7N	8741	$8,741
P6L	21	000021
D12.5	E1234.56	0.123456D+04
I6MDY	20675	02/06/75
A6YMD	750601	75/06/01
A10	HELLO	HELLO

ACTUAL

The ACTUAL attribute describes the type and length of your data as it actually exists in a file. The source of this information is the existing description of the file.

The syntax is

```
ACTUAL = actual
```

where:

actual

Is any of the format types listed in the following table:

ACTUAL Type	Definition
A*n*	Alphanumeric character string (A-Z, 0-9, and other ASCII display characters), where *n* = 1-4,095.
D8	Double-precision floating-point numbers stored internally in eight bytes (D_Floating).
F4	Single-precision floating-point numbers stored internally in four bytes (F_Floating).
I*n*	Binary integers: I1 - Single-byte unsigned, binary integer (unsigned byte) I1.*n* - Single-byte signed, scaled binary integer (signed byte) I2 - 2-byte signed, binary integer (signed word) I2.*n* - 2-byte signed, scaled binary integer (signed word) I4 - 4-byte signed, binary integer (signed longword) I4.*n* - 4-byte signed, scaled binary integer (signed longword) I8 - 8-byte signed, binary integer (signed quadword) I8.*n* - 8-byte signed, scaled binary integer (signed quadword) If an integer field contains an assumed decimal point (meaning it is a scaled integer), the field is represented as I*m*.*n*, where *m* is the total number of storage bytes, and *n* is the number of decimal places for the scaling. For example, I4.1 means a 4-byte number with one decimal place. If I1 (unsigned) is used with negative data values, incorrect values will display. It is important to know if the data will contain negative values and indicate that it is signed by using a scale factor, even if it is simply I1.0 (to indicate single byte signed, no scaling). This rule for signed and unsigned is only unique to ACTUAL I1 and does not apply to I2, I4 or I8 sizes USAGE on a scaled integer may be P, D or I, but should be minimally configured as a utype*m*+2.*n* (for example, I12.2) to account for not having overflows on negative values. However, a USAGE of I is limited to an *m* value of 11. It should also be noted that JDBC JLINK access to a Master File using scaled integers must use USAGEs of P or D for scaling because scaled I is not supported in the JDBC specification. Octaword Integer data type (USAGE=I16) is not supported.

ACTUAL Type	Definition
P*n*	Packed decimal format (packed numeric string) where *n* is the number of bytes (1 to 16), each of which contains two digits, except for the last byte, which contains a digit and the sign. For example, P6 means 11 digits plus a sign, packed two digits to the byte for the total of 6 bytes of storage.
Z*n*	Zoned decimal format (numeric string) where *n* is the number of digits (1 to 31), each of which takes 1 byte of storage. The last byte contains a digit and the sign. There are several standards for zoned data. For Read purposes, only right overpunched standards are supported and can be determined on Read since they are unique. The specific format to use when writing data must be known for read/write purposes. The default is ASCII right overpunch. To change the default, you must edit the EDACONF [.BIN]EDAENV.COM file and add the following logical: `DEFINE /NOLOG IBI_ZONED_OUT_TYPE {1|2}` where: 1 Is the ASCII right overpunched standard (default). 2 Is the EBCDIC right overpunched standard. Zoned right separate numeric or zoned left overpunched numeric formats are not supported. Zoned left separate is supported by using a A*m*+1 ACTUAL and a numeric USAGE format.

ACTUAL Type	Definition
DATE	DATE indicates that the field is an OpenVMS 64-bit datetime stamp. This usage also requires an A4 filler field immediately following in the Master File. DATE/TIMESTAMP. Full OpenVMS date and timestamps are not directly supported. However, using a DEFINE in the Master File, you can declare a field as USAGE=A8, ACTUAL=A8. and then use the virtual (DEFINEd) field with HINPUT ( ) and CVTSTIME ( ) to support HDATE formats for output purposes.
H	Date-Time stamp data that uses formatting options to display raw data or formatted time data, date data, or date-time data. See the *Describing Data With WebFOCUS Language* manual for a full list of options. By default, the values used in this feature are the FOCUS internal time keeping system, but the setting: `SET VMSTIMESTAMP = VMS` Lets actual data based on the OpenVMS Native 64-bit DEC Date specification be read and handled seamlessly. **Note:** Releases prior to 7.7.07 required declaration of DEC Date as A8, and the use of the HINPUT()/CVTSTIMES() functions, which was less than seamless. It is recommended that any such metadata be converted to use H dates and this setting.

The following conversions from ACTUAL format to USAGE (display) format are permitted:

ACTUAL	USAGE
A	A, D, F, I, P, date format
D	D
DATE	date format
F	F
I	I,P date format

ACTUAL	USAGE
P	P, date format
Z	D, F, I, P

The following table shows the USAGE and ACTUAL formats for COBOL, FORTRAN, PL1, and Assembler field descriptions.

COBOL USAGE FORMAT	BYTES OF COBOL PICTURE	INTERNAL STORAGE	ACTUAL FORMAT	USAGE FORMAT
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP*	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1**	"	4	F4	F6
COMP-2***	"	8	D8	D15
COMP-3	9	8	P1	P1
COMP-3	S9V99	8	P2	P5.2
COMP-3	9(4)V9(3)	8	P4	P8.3

COBOL USAGE FORMAT	BYTES OF COBOL PICTURE	INTERNAL STORAGE	ACTUAL FORMAT	USAGE FORMAT
FIXED BINARY(7) (COMP-4)	B or XL1	8	I4	I7

* Equivalent to INTEGER in FORTRAN, FIXED BINARY(31) in PL/1, and F in Assembler.

** Equivalent to REAL in FORTRAN, FLOAT(6) in PL/1, and E in Assembler.

*** Equivalent to DOUBLE PRECISION or REAL*8 in FORTRAN, FLOAT(16) in PL/1, and D in Assembler.

Note:

- The USAGE lengths shown are minimum values. They may be larger if desired. Additional edit options may also be added.
- In USAGE formats, an extra character position is required for the minus sign if negative values are expected.
- PICTURE clauses are not permitted for internal floating-point items.
- USAGE length should allow for maximum possible number of digits.
- In USAGE formats, an extra character position is required for the decimal point.

Describing Indexed Files (SUFFIX=RMS)

The keys within an RMS keyed (indexed) file may be described using a GROUP declaration containing FIELDNAME entries with ALIAS values to describe the contiguous of discontiguous keys. For informations, see *Describing Keys* on page 2059.

Segment Name for Indexed Files

The segment name (SEGNAME value) of the first segment in an indexed file (SUFFIX=RMS) must be ROOT. The remaining segments can have any valid segment name. The only exception to this rule is for unrelated record types where the first segment name value must be DUMMY.

Describing Keys

The primary key is defined by the GROUP attribute and an alias value of KEY in the Master File. If there is a secondary key consisting of more than one field, it must also be described by the GROUP attribute and a numbered key.

The primary key of an RMS file is defined using the GROUP attribute, consisting of one or more fields. A file might only have one keyfield, but it must still be described with the GROUP declaration. The GROUP must contain ALIAS=KEY. Coding KEY without ALIAS= is not sufficient.

The GROUP declaration has the following syntax

```
GROUP=keyname, ALIAS=KEY, USAGE=usage, ACTUAL=actual,$
```

where:

keyname

Is a name of 1 to 66 characters.

The secondary keys of an RMS file are indicated by using ALIAS=KEY(n) in the GROUP or FIELD declaration for the key. For example

```
GROUP=keyname, ALIAS=KEYn, USAGE=usage, ACTUAL=actual,$
```

where:

keyname

Is a name of 1 to 48 characters.

KEY*n*

Indicates the alternate key.

The first alternate key is designated as KEY1, the second as KEY2, and so on.

Alternatively, if the secondary key is made up of only one field, the following syntax can be used:

```
FIELD=keyname, ALIAS=KEYn, USAGE=usage, ACTUAL=actual,$
```

GROUP (for Contiguous Keys)

Contiguous keys consist of adjacent fields.

The GROUP attribute is used to define a contiguous primary key or a secondary key that is comprised of more than one field. The syntax is

```
GROUP=groupname,ALIAS=KEY[n],USAGE=usage,ACTUAL=actual,$
  FIELD=fieldname,ALIAS=alias,USAGE=usage,ACTUAL=actual,$
  .
  .
  .
  FIELD=fieldname,ALIAS=alias,USAGE=usage,ACTUAL=actual,$
```

where:

groupname

Is a name of 1 to 48 characters.

KEY[*n*]

Indicates a contiguous key. Use only KEY to specify a primary key. Use KEY[*n*] to specify a secondary key, where *n* is a number from 1 to 254 that indicates the key reference number.

usage

Is the data type and length designation.

actual

Is the data type and length designation.

For example, consider the contiguous key in the first part of the following Master File:

```
FILENAME=MANUALS,SUFFIX=RMS,$
  SEGMENT=ROOT,SEGTYPE=S0,$
  GROUP=MDOCNUM         ,ALIAS=KEY  ,USAGE=A9      ,ACTUAL=A9      ,$
    FIELDNAME=DOCNUM    ,ALIAS=DN   ,USAGE=A5      ,ACTUAL=A5      ,$
    FIELDNAME=CODE      ,ALIAS=CD   ,USAGE=I6      ,ACTUAL=I4      ,$
    FIELDNAME=MRELEASE  ,ALIAS=MR   ,USAGE=A7      ,ACTUAL=A7      ,$
    FIELDNAME=MPAGES    ,ALIAS=MP   ,USAGE=I5      ,ACTUAL=I2      ,$
         .
         .
         .
```

GROUP (for Discontiguous Keys)

Discontiguous keys consist of non-adjacent fields. If the GROUP attribute is used with discontiguous keys, the syntax is

```
GROUP=groupname,ALIAS=DKEY[n],USAGE=usage,ACTUAL=actual,$
   FIELD=      ,ALIAS=alias  ,USAGE=usage,ACTUAL=actual,$
   .
   .
   .
   FIELD=      ,ALIAS=alias  ,USAGE=usage,ACTUAL=actual,$
```

where:

DKEY[*n*]

Indicates that this GROUP is the key layout for a discontiguous key. The GROUP declaration must explicitly specify ALIAS=DKEY. Use only DKEY to specify a primary key. Use DKEY[*n*] to specify a secondary key, where *n* is a number from 1 to 254 that indicates the key reference number.

alias

Is the name of the base field to which the discontiguous key field corresponds. Note that the field name for the key field must be blank.

usage

Is the data type and length designation.

actual

Is the data type and length designation.

The fields that comprise the discontiguous key are described in the order in which they appear in the record. They are then re-described using the GROUP attribute. The order within the GROUP is determined by their order of significance within the discontiguous key.

For example, consider the discontiguous keys in the first part of the following Master File:

```
FILENAME=MANUALS,SUFFIX=RMS,$
   SEGMENT=ROOT,SEGTYPE=S0,$
      FIELDNAME=DOCNUM    ,ALIAS=          ,USAGE=A5    ,ACTUAL=A5       ,$
      FIELDNAME=CODE      ,ALIAS=CD        ,USAGE=I6    ,ACTUAL=I4       ,$
      FIELDNAME=MRELEASE  ,ALIAS=          ,USAGE=A7    ,ACTUAL=A7       ,$
      FIELDNAME=MPAGES    ,ALIAS=          ,USAGE=I5    ,ACTUAL=I2       ,$
   GROUP=MANUALS_KEY      ,ALIAS=DKEY      ,USAGE=A12   ,ACTUAL=A12      ,$
      FIELDNAME=          ,ALIAS=DOCNUM    ,USAGE=A5    ,ACTUAL=A5       ,$
      FIELDNAME=          ,ALIAS=MRELEASE  ,USAGE=A7    ,ACTUAL=A7       ,$
      .
      .
      .
```

Note that within each segment, you can only have one instance of a key reference number. For example, consider the following: KEY5 and DKEY4 can be in the same segment because they do not reference the same key, but KEY4 and DKEY4 cannot be in the same segment because they reference the same key.

A GROUP of DKEY can occur only at the end of the segment, following all of the "real" field definitions since DKEY contains references to "real" fields as base fields.

Note:

- OCCURS segments cannot contain any key definitions. Non-OCCURS segments must contain either KEY or DKEY definitions.
- Prior editions of this manual noted the use of INDEX=I on primary keys in error. While the use of INDEX=I is relevant to some non-RMS adapters, it is not relevant to the RMS adapter. The actual use of INDEX=I syntax for RMS is still allowed, however, to prevent upward compatibility problems with existing metadata, but it serves no actual function.

USAGE and ACTUAL

For multi-field GROUPs, USAGE and ACTUAL formats are always alphanumeric. The ACTUAL attribute is A*n*, where *n* is the sum of the actual lengths of the subordinate fields. The USAGE format is the sum of the internal storage lengths of the subordinate fields.

- Fields of USAGE I have an internal storage length of 4.
- Fields of USAGE F have an internal storage length of 4.
- Fields of USAGE P have an internal storage length of either 8 or 16.
- Fields of USAGE D have an internal storage length of 8.
- Alphanumeric fields have an internal storage length equal to the number of characters they contain as their field length. For example, fields of type A*n* have an internal storage length of *n*.
- Natural date formats that are treated as integers have an internal storage length of 4.

For example, consider the discontiguous keys in a part of the following Master File:

```
.
.
.
FIELDNAME=FIELD1   ,ALIAS=          ,USAGE=P6   ,ACTUAL=P2       ,$
FIELDNAME=FIELD2   ,ALIAS=          ,USAGE=I9   ,ACTUAL=I4       ,$
FIELDNAME=FIELD3   ,ALIAS=          ,USAGE=A2   ,ACTUAL=A2       ,$
GROUP=ALTERNATE    ,ALIAS=DKEY      ,USAGE=A10  ,ACTUAL=A4       ,$
FIELDNAME=         ,ALIAS=FIELD3    ,USAGE=A2   ,ACTUAL=A2       ,$
FIELDNAME=         ,ALIAS=FIELD1    ,USAGE=P6   ,ACTUAL=P2       ,$
.
.
.
```

The GROUP declaration USAGE attribute tells how many positions to use for the group key. If this length is wrong, the group key will not be used correctly.

In this example, the lengths of the ACTUAL attributes for subordinate fields FIELD3 and FIELD1 total 4, which is the length of the ACTUAL attribute of the GROUP key. The lengths of the USAGE attributes for the subordinate fields total 8. However, the length of the GROUP key USAGE attribute is found by adding their internal storage lengths as specified by the field types: 2 for USAGE=A2 and 8 for USAGE=P6, for a total of 10.

Single-Field Secondary Keys

Single-field secondary keys must be described as fields whose ALIAS must be the key reference number as described in the RMS File Description Language (FDL), that is, KEY*n* or DKEY*n*, where *n* is a number from 1 to 254. Secondary keys can be described as GROUPs if they consist of portions with dissimilar formats. For example,

```
FILENAME=CUST,SUFFIX=RMS,$
SEGNAME=ROOT,SEGTYPE=S0,$
  GROUP=G             ,ALIAS=KEY    ,USAGE=A10  ,ACTUAL=A10,$
    FIELDNAME=SSN     ,ALIAS=SSN    ,USAGE=A10  ,ACTUAL=A10,$
    FIELDNAME=FNAME   ,ALIAS=KEY1   ,USAGE=A10  ,ACTUAL=A10,$
    FIELDNAME=LNAME   ,ALIAS=KEY2   ,USAGE=A10  ,ACTUAL=A10,$
```

Here, SSN is a primary key and FNAME and LNAME are secondary keys.

If you are not sure of the secondary keys associated with a given file, you can use the OpenVMS ANALYSE facility. Refer to the OpenVMS documentation for ANALYSE/RMS. The DIRECTORY command with the /FULL qualifier will also show how many keys the file has.

Describing Complex RMS Keyed Files

These topics discuss various ways of describing complex RMS keyed files.

Describing Multiple Record Types

Files may have records that must be deciphered according to a record type indicator in the record itself. Describing these files involves two things. First, each different record type will need its own segment. Second, the relationship between the different record types (that is, between the different segments) will need to be determined and expressed using the PARENT attribute for each segment.

Using RECTYPE

The syntax for defining a RECTYPE field is

```
FIELDNAME=RECTYPE,ALIAS=alias,USAGE=usage,ACTUAL= actual
 [,ACCEPT=list/range],$
```

where:

`RECTYPE`

Is the required field name.

`alias`

Is the primary RECTYPE identifier. If there is an ACCEPT list or range, this value is any valid alias name.

list

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If an item in the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single RECTYPE value.

For example:

```
FIELDNAME=RECTYPE, ALIAS=A, USAGE=A1, ACTUAL=A1, ACCEPT=A OR B OR C,$
```

range

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If either value contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

For example:

```
FIELDNAME=RECTYPE,ALIAS=100,USAGE=P3,ACTUAL=P2,ACCEPT=70 TO 100,$
```

When using an ACCEPT list or range, also called generalized RECTYPE, the RECTYPE field name is not unique across segments, therefore it must not be used in a WHERE clause of an SQL request. It is the responsibility of the client application to determine if the row for the RECTYPE is required. If the client application cannot do this, you must code individual segments for each RECTYPE.

To illustrate the use of the generalized RECTYPE capability in RMS file descriptions, consider the following record layouts in the DOC file. Record type DN is the root segment and contains the document number and title. Record types M, I, and C contain information about manuals, installation guides, and course guides, respectively. Notice that record types M and I have the same layout.

Record Type DN:

```
---KEY---
+-----------------------------------------------------------------+
DOCID    FILLER   RECTYPE   TITLE
+-----------------------------------------------------------------+
```

Record Type M:

```
--------KEY------
+-------------------------------------------------------------------+
MDOCID   MDATE   RECTYPE   MRELEASE            MPAGES     FILLER
+-------------------------------------------------------------------+
```

Record Type I:

```
--------KEY------
+-------------------------------------------------------------------+
IDOCID   IDATE   RECTYPE   IRELEASE            IPAGES     FILLER
+-------------------------------------------------------------------+
```

Record Type C:

```
--------KEY------
+-------------------------------------------------------------------+
CRSEDOC  CDATE   RECTYPE   COURSENUM   LEVEL   CPAGES     FILLER
+-------------------------------------------------------------------+
```

Without the ACCEPT attribute, each of the four record types must be described as separate segments in the Master File. For example, a unique set of field names must be provided for record type M and for record type I, although they have the same layout.

The generalized RECTYPE capability enables you to code just one set of field names that apply to the record layout for both record type M and record type I. The ACCEPT attribute can be used for any RECTYPE specification, even when there is only one acceptable value.

```
FILENAME=DOC, SUFFIX=RMS,$
SEGNAME=ROOT, SEGTYPE=S0,$
 GROUP=DOCNUM,  ALIAS=KEY,          A5,  A5, $
  FIELD=DOCID,      ALIAS=,         A5,  A5, $
 FIELD=FILLER,      ALIAS=,         A5,  A5, $
 FIELD=RECTYPE,     ALIAS=,         A3,  A3, $
 FIELD=TITLE,       ALIAS=,         A18, A18,$
SEGNAME=MANUALS, PARENT=ROOT, SEGTYPE=S0,    $
 GROUP=MDOCNUM, ALIAS=KEY,          A10, A10,$
  FIELD=MDOCID,     ALIAS=,         A5,  A5, $
  FIELD=MDATE,      ALIAS=,         A5,  A5, $
 FIELD=RECTYPE,     ALIAS=M,        A3,  A3, ACCEPT = M OR I,$
 FIELD=MRELEASE,    ALIAS=,         A7,  A7, $
 FIELD=MPAGES,      ALIAS=,         I5,  A5, $
 FIELD=FILLER,      ALIAS=,         A6,  A6, $
SEGNAME=COURSES, PARENT=ROOT, SEGTYPE=S0,    $
 GROUP=CRSEDOC, ALIAS=KEY,          A10, A10,$
  FIELD=CDOCID,     ALIAS=,         A5,  A5, $
  FIELD=CDATE,      ALIAS=,         A5,  A5, $
 FIELD=RECTYPE,     ALIAS=C,        A3,  A3, $
 FIELD=COURSENUM,   ALIAS=,         A4,  A4, $
 FIELD=LEVEL,       ALIAS=,         A2,  A2, $
 FIELD=CPAGES,      ALIAS=,         I5,  A5, $
 FIELD=FILLER,      ALIAS=,         A7,  A7, $
```

Describing Related Record Types

Consider the LIBRARY file that contains three types of records, related by a combination of the key and the RECTYPE attribute. The ROOT records have a key that consists of the publisher's number. The BOOKINFO segment has a key that consists of that same publisher's number, plus a hard or soft-cover indicator. The SERIANO key consists of the first two elements, plus a record type.

In the sample file, the repetition of the publisher number interrelates the three types of records. The Master File for this file would look like the following:

```
FILENAME=LIBRARY6,SUFFIX=RMS,$
SEGNAME=ROOT,SEGTYPE=S0,$
 GROUP=PUBKEY          ,ALIAS=KEY   ,USAGE=A10     ,ACTUAL=A10    ,$
  FIELDNAME=PUBNO      ,ALIAS=PN    ,USAGE=A10     ,ACTUAL=A10    ,$
  FIELDNAME=FILLER     ,ALIAS=      ,USAGE=A1      ,ACTUAL=A1     ,$
  FIELDNAME=RECTYPE    ,ALIAS=1     ,USAGE=A1      ,ACTUAL=A1     ,$
  FIELDNAME=AUTHOR     ,ALIAS=AT    ,USAGE=A25     ,ACTUAL=A25    ,$
  FIELDNAME=TITLE      ,ALIAS=TL    ,USAGE=A50     ,ACTUAL=A50    ,$
SEGNAME=BOOKINFO,SEGTYPE=S0,PARENT=ROOT,$
 GROUP=BOINKEY         ,ALIAS=KEY   ,USAGE=A11     ,ACTUAL=A11    ,$
  FIELDNAME=PUBNO1     ,ALIAS=P1    ,USAGE=A10     ,ACTUAL=A10    ,$
  FIELDNAME=BINDING    ,ALIAS=BI    ,USAGE=A1      ,ACTUAL=A1     ,$
  FIELDNAME=RECTYPE    ,ALIAS=2     ,USAGE=A1      ,ACTUAL=A1     ,$
  FIELDNAME=PRICE      ,ALIAS=PR    ,USAGE=D8.2N   ,ACTUAL=D8     ,$
SEGNAME=SERIANO,SEGTYPE=S0,PARENT=BOOKINFO,$
 GROUP=SERIKEY      ,   ALIAS=KEY   ,USAGE=A12     ,ACTUAL=A12    ,$
  FIELDNAME=PUBNO2   ,  ALIAS=P2    ,USAGE=A10     ,ACTUAL=A10    ,$
  FIELDNAME=BINDING1   ,ALIAS=B1    ,USAGE=A1      ,ACTUAL=A1     ,$
  FIELDNAME=RECTYPE    ,ALIAS=3     ,USAGE=A1      ,ACTUAL=A1     ,$
  FIELDNAME=SERIAL     ,ALIAS=SN    ,USAGE=A15     ,ACTUAL=A15    ,$
SEGNAME=SYNOPSIS,SEGTYPE=S0,PARENT=ROOT,OCCURS=VARIABLE,$
  FIELDNAME=PLOTLINE   ,ALIAS=PLOTL ,USAGE=A10     ,ACTUAL=A10    ,$
```

A typical query might request information on price and call numbers for a specific publisher's number:

```
PRINT PRICE AND SERIAL BY PUBNO
IF PUBNO EQ 1234567890 OR 9876054321
```

Since PUBNO is part of the key, the retrieval can be made and the processing continues. For greater speed retrieval, you could add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Record Types

A file may contain records that are not related to each other. Records with varying RECTYPEs exist independently of each other in a file, and the sequence of records in the file may be random.

Consider our LIBRARY file. Suppose that the file has three types of records: book information, magazine information, and newspaper information.

Since book information, magazine information, and newspaper information have nothing in common, these record types cannot be described in a parent/child relationship.

The records simply look like the following:

```
BOOK     MAGAZINE     NEWSPAPER
```

To describe a file with unrelated records, you must make the record types descendants of a root segment named DUMMY.

The following rules apply to the DUMMY segment:

- The root (top) segment name must be DUMMY.
- It can have only one field, with an empty FIELDNAME and ALIAS.
- Both its USAGE and ACTUAL attributes must be A1.

All of the other segments must be descendants of the DUMMY segment.

The Master File for this file would look like the following:

```
FILENAME=LIBRARY3, SUFFIX=FIX,$
SEGMENT=DUMMY,SEGTYPE=S0,$
   FIELDNAME=            ,ALIAS=       ,USAGE=A1      ,ACTUAL=A1      ,$


SEGMENT=BOOK,SEGTYPE=S0,PARENT=DUMMY,$
   FIELDNAME=RECTYPE     ,ALIAS=B      ,USAGE=A1      ,ACTUAL=A1      ,$
   GROUP=PUBNUM          ,ALIAS=KEY    ,USAGE=AI0     ,ACTUAL=A10     ,$
   FIELDNAME=PUBNO       ,ALIAS=PN     ,USAGE=A10     ,ACTUAL=A10     ,$
   FIELDNAME=AUTHOR      ,ALIAS=AT     ,USAGE=A25     ,ACTUAL=A25     ,$
   FIELDNAME=TITLE       ,ALIAS=TL     ,USAGE=A50     ,ACTUAL=A50     ,$
   FIELDNAME=BINDING     ,ALIAS=BI     ,USAGE=A1      ,ACTUAL=A1      ,$
   FIELDNAME=PRICE       ,ALIAS=PR     ,USAGE=D8.2N   ,ACTUAL=D8      ,$
   FIELDNAME=SERIAL      ,ALIAS=SN     ,USAGE=A15     ,ACTUAL=A15     ,$
   FIELDNAME=SYNOPSIS    ,ALIAS=SY     ,USAGE=A150    ,ACTUAL=A150    ,$


SEGMENT=MAGAZINE,SEGTYPE=S0,PARENT=DUMMY,$
   FIELDNAME=RECTYPE     ,ALIAS=M      ,USAGE=A1      ,ACTUAL=A1      ,$
   GROUP=PERNUM          ,ALIAS=KEY    ,USAGE=A10     ,ACTUAL=A10     ,$
   FIELDNAME=PER_NO      ,ALIAS=PN     ,USAGE=A10     ,ACTUAL=A10     ,$
   FIELDNAME=PER_NAME    ,ALIAS=NA     ,USAGE=A50     ,ACTUAL=A50     ,$
   FIELDNAME=VOL_NO      ,ALIAS=VN     ,USAGE=I2      ,ACTUAL=I2      ,$
   FIELDNAME=ISSUE_NO    ,ALIAS=IN     ,USAGE=I2      ,ACTUAL=I2      ,$
   FIELDNAME=PER_DATE    ,ALIAS=DT     ,USAGE=YYMD    ,ACTUAL=DATE    ,$
   FIELDNAME=            ,ALIAS=FILLER ,USAGE=A4      ,ACTUAL=A4      ,$
```

```
SEGMENT=NEWSPAP,SEGTYPE=S0,PARENT=DUMMY,$
   FIELDNAME=RECTYPE     ,ALIAS=N      ,USAGE=A1      ,ACTUAL=A1      ,$
   FIELDNAME=NEW_NAME    ,ALIAS=NN     ,USAGE=A50     ,ACTUAL=A50     ,$
   FIELDNAME=NEW_DATE    ,ALIAS=ND     ,USAGE=I6MDY   ,ACTUAL=I4      ,$
   FIELDNAME=            ,ALIAS=FILLER ,USAGE=A4      ,ACTUAL=A4      ,$
   GROUP=PAPVI           ,ALIAS=KEY    ,USAGE=A4      ,ACTUAL=A4      ,$
   FIELDNAME=NVOL_NO     ,ALIAS=NV     ,USAGE=I2      ,ACTUAL=I2      ,$
   FIELDNAME=NISSUE      ,ALIAS=NI     ,USAGE=I2      ,ACTUAL=I2      ,$
```

Describing Embedded Repeating Data

Some records may contain embedded repeating data. Consider the following record layout:

```
A   B   C1   C2   C1   C2
```

Fields C1 and C2 repeat within this data record. C1 has an initial value, as does C2. C1 then provides a second value for that field, as does C2. Thus, there are two values for fields C1 and C2 for every one value for fields A and B.

The number of times C1 and C2 occur does not have to be fixed, depending on the value of a counter field. Suppose field B is this counter field. In the case shown above, the value of field B is 2, since C1 and C2 occur twice. The value of field B in the next record may be different, and fields C1 and C2 will occur that number of times.

The number of times fields C1 and C2 occur can also be variable. In this case, everything after fields A and B is assumed to be a series of C1s and C2s, alternating to the end of the record.

You describe these multiply occurring fields by placing them in a separate segment. Fields A and B are placed in the first segment, called the root segment. Fields C1 and C2, which occur multiple times in relation to A and B, are placed in a descendant segment. You use an additional segment attribute, the OCCURS attribute, to specify that this segment is a multiply occurring segment.

Repeating fields or groups of fields described by the OCCURS attribute are not supported for free-format (comma-delimited) sequential files.

OCCURS

The OCCURS attribute is an optional segment attribute used to describe records containing repeating fields or groups of fields. You define such records by describing the singly occurring fields in one segment and the multiply occurring fields in another, subordinate segment. The OCCURS attribute appears in the declaration for the subordinate segment.

The syntax is

```
OCCURS = {n|fieldname|VARIABLE},$
```

where:

n

Is an integer value showing the number of occurrences (1 to 4095).

fieldname

Names a data field in the parent segment, that is, a counter specifying the number of occurrences of the descendant segment.

VARIABLE

Indicates that the number of occurrences varies from record to record. The number of occurrences is computed from the record length (that is, if the field lengths for the segment add up to 40, and 120 characters are read in, it means there are three occurrences).

When different types of records are combined in one file, each record type can contain only one segment (defined as OCCURS=VARIABLE). It may have OCCURS descendants (if it contains a nested group), but it may not be followed by any other segment with the same parent, that is, there can be no other segments to its right in the structure. This restriction is necessary to ensure that data in the record is interpreted correctly.

Example: Using the OCCURS Attribute

You place the OCCURS attribute in the segment declaration after the PARENT attribute. Consider the following record layout:

```
A   B   C1   C2   C1   C2
```

You have two occurrences of fields C1 and C2 for every one occurrence of fields A and B. Thus, to describe this file, you place fields A and B in the root segment, and fields C1 and C2 in the descendant segment.

You describe this file with the following Master File:

```
FILENAME=EXAMPLE1,SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2     ,$
  FIELDNAME=A     ,ALIAS=    ,USAGE=A2    ,ACTUAL=A2    ,$
  FIELDNAME=B     ,ALIAS=    ,USAGE=A1    ,ACTUAL=A1    ,$
 SEGNAME=TWO,PARENT=ONE,OCCURS=2,SEGTYPE=S0,$
  FIELDNAME=C1    ,ALIAS=    ,USAGE=I4    ,ACTUAL=I2    ,$
  FIELDNAME=C2    ,ALIAS=    ,USAGE=I4    ,ACTUAL=I2    ,$
```

Note: OCCURS is not supported for Write Access. However, if the fields occur a specific number of times, an alternate Master File can be built with the fields described that number of times (for example, PAYMENT_1, PAYMENT_2 if Payment occurs two times). Using the alternate Master, specific instances of the OCCURS can be referenced. If the OCCURS segments will not be referenced during the write, the alternate Master is not needed. However, the message (FOC1305) RMS Duplicate Record will display.

Example: Describing Parallel and Nested Sets of OCCURS Segments

You can have several sets of repeating fields in your data structure. You describe each of these sets of fields as a separate segment in your Master File.

Sets of repeating fields can be divided into two basic types: parallel and nested. Parallel sets of repeating fields are unrelated (that is, they have no parent/child or logical relationship). Consider the following record layout:

```
A1   A2   B1   B2   B1   B2   C1   C2   C1   C2   C1   C2
```

In this example, fields B1 and B2 and fields C1 and C2 repeat within the record. The number of times that fields B1 and B2 occur is unrelated to the number of times fields C1 and C2 occur. Fields B1 and B2 and fields C1 and C2 are parallel sets of repeating fields. They should be described in the Master File as children of the same parent, the segment that contains fields A1 and A2. The following Master File illustrates this relationship:

```
FILENAME=EXAMPLE1, SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2      ,$
  FIELDNAME=A1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
  FIELDNAME=A2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
 SEGNAME=TWO, SEGTYPE=S0, PARENT=ONE,     OCCURS=2,     $
  FIELDNAME=B1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
  FIELDNAME=B2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
 SEGNAME=THREE, SEGTYPE=S0, PARENT=ONE,   OCCURS=3,     $
  FIELDNAME=C1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
  FIELDNAME=C2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,    $
```

Nested sets of repeating fields are those whose occurrence in some way depends on one another. Consider the following data structure:

```
A1   A2   B1   B2   C1   C1   B1   B2   C1   C1   C1
```

In this example, field C1 occurs after fields B1 and B2 occur once. Field C1 occurs a varying number of times, as recorded by a counter field, B2. There will not be a set of occurrences of C1 unless C1 is preceded by an occurrence of fields B1 and B2. Fields B1, B2, and C1 are a nested set of repeating fields. The following Master File illustrates this relationship:

```
FILENAME=EXAMPLE2, SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2         ,$
  FIELDNAME=A1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
  FIELDNAME=A2,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
 SEGNAME=TWO,       SEGTYPE=S0, PARENT=ONE, OCCURS=2,       $
  FIELDNAME=B1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
  FIELDNAME=B2,    ALIAS= ,    USAGE=I6,    ACTUAL=I4,       $
 SEGNAME=THREE,     SEGTYPE=S0, PARENT=TWO, OCCURS=B2,      $
  FIELDNAME=C1,    ALIAS= ,    USAGE=A2,    ACTUAL=A2,       $
```

Since field C1 repeats with relation to fields B1 and B2, which repeat in relation to fields A1 and A2, field C1 is described as a separate, descendant segment of Segment TWO, which is in turn a descendant of Segment ONE.

The following data structure contains both nested and parallel sets of repeating fields:

```
A1 A2  B1  B2  C1  C1  C1  B1  B2  C1  C1  C1  C1  D1  D1  E1  E1  E1  E1
```

You describe this with the Master File that follows. The PARENT attribute is used to describe the logical relationship of the segments. Note that the assignment of the PARENT attribute shows you how the occurrences are nested:

```
FILENAME=EXAMPLE3,SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2         ,$
  FIELDNAME=A1    ,ALIAS=    ,USAGE=A2    ,ACTUAL=A2    ,    $
  FIELDNAME=A2    ,ALIAS=    ,USAGE=I4    ,ACTUAL=I1    ,    $
 SEGNAME=TWO,PARENT=ONE,OCCURS=2,SEGTYPE=S0,                  $
  FIELDNAME=B1    ,ALIAS=    ,USAGE=A15   ,ACTUAL=A15   ,    $
  FIELDNAME=B2    ,ALIAS=    ,USAGE=I4    ,ACTUAL=I1    ,    $
 SEGNAME=THREE,PARENT=TWO,OCCURS=B2,SEGTYPE=S0,               $
  FIELDNAME=C1    ,ALIAS=    ,USAGE=A25   ,ACTUAL=A25   ,    $
 SEGNAME=FOUR,PARENT=ONE,OCCURS=A2,SEGTYPE=S0,                $
  FIELDNAME=D1    ,ALIAS=    ,USAGE=A15   ,ACTUAL=A15   ,    $
 SEGNAME=FIVE,PARENT=ONE,OCCURS=VARIABLE,SEGTYPE=S0,          $
  FIELDNAME=E1    ,ALIAS=    ,USAGE=A5    ,ACTUAL=A5    ,    $
```

Note:

- Segments ONE, TWO, and THREE represent a nested group of repeating segments. Fields B1 and B2 occur a fixed number of times, so OCCURS equals 2. Field C1 occurs a certain number of times within each occurrence of fields B1 and B2. The number of times C1 occurs is determined by the value of B2, which is a counter. The value of B2 is 3 for the first occurrence of Segment TWO, and 4 for the second occurrence.
- Segments FOUR and FIVE consist of fields that repeat independently within the parent segment. They have no relationship to each other or to Segment TWO except for the common parent, so they represent a parallel group of repeating segments.
- As in the case of Segment THREE, the number of times Segment FOUR occurs is determined by a counter in its parent, A2. In this example, the value of A2 is 2.

- The number of times Segment FIVE occurs is VARIABLE. The rest of the fields in the record (all those to the right of the first occurrence of E1) as occurrences of E1. To ensure that data in the record is interpreted correctly, a segment defined as OCCURS=VARIABLE must be at the end of the record. In a data structure diagram, it will be the right-most segment. There can only be one segment defined as OCCURS=VARIABLE for each parent.

POSITION

The POSITION attribute is an optional attribute. It is used to describe a record in which non-repeating data follows embedded repeating data.

You describe the file as a multi-segment structure, made up of a parent segment and at least one child segment that contains the embedded repeating data. The parent segment is made up of whatever singly occurring fields are in the record, as well as an alphanumeric field (or fields) that indicates where the embedded repeating data appear in the record. The alphanumeric field is a placeholder that is the exact length of the combined embedded repeating data. For example, if you have four occurrences of an 8-character field, the length of the placeholder in the parent segment will be 32 characters.

The POSITION attribute is described in the child segment. It gives the name of the placeholder field in the parent segment.

The syntax is

```
POSITION = fieldname
```

where:

fieldname

Is the name of the field in the parent segment that defines the starting position of the multiply occurring fields.

Consider the following record layout:

```
A1   Q1   Q1   Q1   Q1   A2   A3   A4
```

In this example, field Q1 repeats four times in the middle of the record. When you describe this structure, you specify a field that occupies the position of the four Q1 fields in the record. You then assign the actual Q1 fields to a descendant, multiply occurring segment. The POSITION attribute, specified in the descendant segment, gives the name of the place holder field in the parent segment.

The Master File for this file would look like this:

```
FILENAME=EXAMPLE3,SUFFIX=RMS,$
 GROUP=ONE    ,ALIAS=KEY    ,USAGE=A2    ,ACTUAL=A2       ,$
  FIELDNAME=A1      ,ALIAS=    ,USAGE=A2   ,ACTUAL=A2    ,$
  FIELDNAME=QFIL    ,ALIAS=    ,USAGE=A1    ,ACTUAL=A32    ,$
  FIELDNAME=A2      ,ALIAS=    ,USAGE=I2    ,ACTUAL=I2     ,$
  FIELDNAME=A3      ,ALIAS=    ,USAGE=A10   ,ACTUAL=A10    ,$
  FIELDNAME=A4      ,ALIAS=    ,USAGE=A15   ,ACTUAL=A15    ,$
 SEGNAME=TWO,SEGTYPE=S0,PARENT=ONE,POSITION=QFIL,OCCURS=4,$
  FIELDNAME=Q1      ,ALIAS=    ,USAGE=D8    ,ACTUAL=D8     ,$
```

If the total length of the multiple occurrences of the field(s) is greater than 256, you can use a filler field after the place holder field to make up the remaining length. This is because the format of a field cannot exceed 256 bytes.

This structure will only work if you have a fixed number of occurrences of the repeating field. This means that the OCCURS attribute of the descendant segment must be of the type OCCURS=*n*. The following will not work: OCCURS=*fieldname* or OCCURS=VARIABLE.

When a segment is coded with ...OCCURS=*n*, POSITION=*fieldname*, ..., all segments that follow using OCCURS=*n* or OCCURS=*fieldname*, must use the POSITION attribute to correctly map data.

ORDER Field

In an OCCURS segment, the order of the data may be significant. For example, the numbers may represent monthly or quarterly data, but the record itself may not explicitly specify the month (or quarter) to which the data applies.

You can add a special ORDER field to your Master File to identify the individual occurrences of data uniquely within their parent segment. This is typically done to the screen by the occurrence number versus a field value (for example, the month is 12). The sequence number of each instance of the segment is automatically defined as a virtual field, and does not actually exist within the file.

The following syntax rules apply to the ORDER field:

- It must be the last field described in an OCCURS segment. If you are using MAPVALUE, then MAPVALUE must be the last field preceded by the ORDER field.
- The field name is arbitrary.
- The ALIAS attribute must be ORDER.
- The USAGE attribute must be I*n*, with any appropriate edit options; "*n*" is a number from 1 to 9.
- The ACTUAL attribute must be I4.

For example:

```
FIELD=ACT_MONTH, ALIAS=ORDER, USAGE=I2, ACTUAL=I4,$
```

Order values are assigned sequentially (1,2,3,...) and the order value is reset to 1 when a new instance of the parent segment is retrieved. The value is assigned prior to any selection tests that might accept or reject the record, and can be used in a screening condition. For example, to obtain data for the month of June, type:

```
SUM AMOUNT...
IF ORDER IS 6
```

Example: Describing Repeating Embedded Data With Record Type Indicators

If a file contains records that have repeating embedded data, the OCCURS attribute is used to describe a separate segment for the repeating fields. In some files, however, the repeating fields themselves must be identified according to a record type indicator. Suppose you want to describe a file that, schematically, looks like the following:

```
A  Rectype B C     Rectype  B C
A  Rectype D       Rectype  D
```

You would need to describe three segments in your Master File, with A as the root segment, and segments for B, C, and D, as two descendant OCCURS segments for A.

This is its Master File:

```
FILE=ABCD, SUFFIX=RMS,$
 GROUP=ONE             ,ALIAS=KEY      ,ACTUAL=A2   ,USAGE=A2  ,$
  FIELDNAME=A          ,ALIAS=         ,USAGE=A2    ,ACTUAL=A2  ,$
 SEGNAME=BC_SEG, SEGTYPE=S0, PARENT=ONE, OCCURS=2,$
  FIELDNAME=RECTYPE    ,ALIAS=1        ,USAGE=I2    ,ACTUAL=I2   ,$
  FIELDNAME=B          ,ALIAS=         ,USAGE=A5    ,ACTUAL=A5   ,$
  FIELDNAME=C          ,ALIAS=         ,USAGE=A5    ,ACTUAL=A5   ,$
 SEGNAME=D_SEG, SEGTYPE=S0, PARENT=ONE, OCCURS=2,$
  FIELDNAME=RECTYPE    ,ALIAS=2        ,USAGE=I2    ,ACTUAL=I2   ,$
  FIELDNAME=D          ,ALIAS=         ,USAGE=A15   ,ACTUAL=A15  ,$
```

Each of the two descendant OCCURS segments in this example depends on the record type indicator that appears for each occurrence.

All the rules of syntax for using RECTYPE fields and OCCURS segments apply to RECTYPEs within OCCURS segments.

Since the OCCURS segment depends on the RECTYPE indicator for its evaluation, the RECTYPE must appear at the start of each OCCURS segment. This allows very complex files to be described, including those with nested and parallel repeating groups that depend on RECTYPEs. For example:

```
A   Rectype B C   Rectype D   Rectype D   Rectype E   Rectype E
```

In this case, B/C, and D represent a nested repeating group, and E represents a parallel repeating group.

Syntax:

How to Describe Repeating Groups Depending on a Preceding Record Type Indicator Using MAPFIELD/MAPVALUE

MAPFIELD is assigned as the ALIAS of the field that will be the record type indicator. You can give this field any name; it is otherwise described according to the usual syntax:

```
FIELD=name, ALIAS=MAPFIELD, USAGE=usage, ACTUAL=actual,$
```

The descendant segments, whose values depend on the value of the MAPFIELD, are described as separate segments, one for each possible value of MAPFIELD, and all descending from the segment that has the MAPFIELD. A special field, MAPVALUE, is described as the last field in these descendant segments. If an ORDER field is chosen, it must be used before the MAPVALUE. The actual MAPFIELD value is supplied as the MAPVALUE ALIAS.

The syntax is

```
FIELDNAME=MAPVALUE,ALIAS=alias,USAGE=usage{Pn},ACTUAL={Pn},
  ACCEPT=list/range],$
```

where:

`MAPVALUE`

Indicates that the segment depends on a MAPFIELD in its parent segment.

alias

Is the primary MAPFIELD value that identifies the segment. If there is an ACCEPT list, this value is any value in the ACCEPT list or range.

usage

Is the same format as the MAPFIELD format in the parent segment.

actual

Is the same format as the MAPFIELD format in the parent segment.

list

Is a list of one or more lines of specific MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If an item in the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single MAPFIELD value. For example:

```
FIELDNAME=MAPVALUE, ALIAS=A, USAGE=A1, ACTUAL=A1, ACCEPT=A OR B OR C,$
```

range

Is a range of one or more lines of MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If either value in the range contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

In some cases, the record type indicates what kind of repeating data will follow. Schematically, the records would look like the following:

```
A  B  Recordtype  (1)   C D   C D   C D
A  B  Recordtype  (2)   E E
```

The first record contains "header" information, values for A and B, followed by an OCCURS segment of C and D that was identified by its preceding record type indicator. The second record has a different record type indicator and contains a different repeating group, this time for E.

Since the OCCURS segments are identified by the record type indicator, rather than the parent A/B segment, you can use the keyword MAPFIELD. MAPFIELD identifies a field like RECTYPE, but, since the OCCURS segments will each have specific values for MAPFIELD, its value is associated with each OCCURS segment by means of a complementary field named MAPVALUE.

The Master File for this file would look like the following:

```
FILENAME=EXAMPLE,SUFFIX=RMS,$
GROUP=ONE            ,ALIAS=KEY       ,ACTUAL=A2     ,USAGE=A2  ,$
FIELDNAME=A          ,ALIAS=          ,USAGE=A2      ,ACTUAL=A2,$
FIELDNAME=B          ,ALIAS=          ,USAGE=A10     ,ACTUAL=A10,$
FIELDNAME=FLAG       ,ALIAS= MAPFIELD ,USAGE=A1      ,ACTUAL=A1,$
SEGNAME=TWO,SEGTYPE=S0,PARENT=ONE,OCCURS=VARIABLE,$
FIELDNAME=C          ,ALIAS=          ,USAGE=A5      ,ACTUAL=A5,$
FIELDNAME=D          ,ALIAS=          ,USAGE=A7      ,ACTUAL=A7,$
FIELDNAME=MAPVALUE   ,ALIAS=1         ,USAGE=A1      ,ACTUAL=A1,$
SEGNAME=THREE,SEGTYPE=S0,PARENT=ONE,OCCURS=VARIABLE,$
FIELDNAME=E          ,ALIAS=          ,USAGE=D12.2   ,ACTUAL=D8,$
FIELDNAME=MAPVALUE   ,ALIAS=2         ,USAGE=A1      ,ACTUAL=A1,$
```

Associating an RMS Data Source to a Master File

You can associate an RMS file to a Master File using the following methods:

- **DATASET attribute in a Master File.** The DATASET attribute in a Master File specifies the physical or app location for the data source to be allocated. This attribute is used at the file declaration level of the Master File, as described in *DATASET* on page 2048. The synonym creation facility uses the DATASET attribute in the Master File. For details, see *How to Specify a Physical File Location Using DATASET= and the RMS Data Source* on page 2078.
- **RMSFILE attribute in an Access File.** You can use either the RMSFILE attribute or a DATESET attribute if you are creating Master and Access Files from scratch using a text editor. If both DATASET and RMSFILE are declared, DATASET takes precedence. For details, see *How to Associate an RMS Data Source in an Access File* on page 2079.
- **FILEDEF command.** An explicit allocation (FILEDEF) for the data file is checked for when a DATASET or RMSFILE attribute is detected in a Master File. If an explicit allocation is found, the DATASET and RMSFILE attributes are ignored and the allocation is used.

  If the data file name is not allocated with a FILEDEF, an internal command is issued to perform the allocation using the DATASET or RMSFILE values. This allocation is stored temporarily and is released when a new Master File is used or when the session terminates.

If both attributes exist, DATASET takes precedence over RMSFILE. However, an explicit FILEDEF command takes precedence over both the DATASET and RMSFILE attributes.

Syntax: How to Specify a Physical File Location Using DATASET= and the RMS Data Source

The DATASET attribute is used at the file declaration level of the Master File. The syntax is

```
{DATASET|DATA}={'filename'|app/filename.[extension]}
```

where:

filename

Is the platform-dependent physical name of the data source. In a prior release, the physical name was required to be in single quotes. This is still valid syntax, but not required. A direct logical name such as MYQ2DATA: is also valid but must include the colon character, so it is distinct from an app-based name.

app/filename[*.extension*]

Is an app-based logical name of the data source. The default extension when no extension is supplied is .dat.

Note: If a DATASET allocation is in effect, a CHECK FILE command must be issued in order to override it by an explicit allocation command. The CHECK FILE command will deallocate the allocation created by DATASET. The default extension when no extension is supplied is .dat.

Example: Allocating an RMS Data Source Using the DATASET Attribute

The following example illustrates how to allocate an RMS data source on the file declaration level and for an alternate index:

```
FILE=EXERVSM1, SUFFIX=RMS,$
DATASET='MYDISK:[mydata.sample]testdata.dat',$
SEGNAME=ROOT , SEGTYPE=S0,$
 GROUP=KEY1  , ALIAS=KEY , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD1 , ALIAS=F1  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD2 , ALIAS=F2  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD3 , ALIAS=DD1 , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD4 , ALIAS=F4  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD5 , ALIAS=F5  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD6 , ALIAS=F6  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD7 , ALIAS=F7  , FORMAT=A4, ACTUAL=A4 ,$
```

Syntax: How to Associate an RMS Data Source in an Access File

The Access File has the following format

```
RMSFILE   = {'filename'|app/filename.[extension]},
[ACCESS   = {SHARED|READONLY},]
[LOCKMODE = {SKIP|KEEP} ,]
[STATMODE = {EXCEPTIONS|ON|OFF},] $
```

where:

filename

Is the platform-dependent physical name of the data source. In a prior release, the physical name was required to be in single quotes. This is still valid syntax, but not required. A direct logical name such as MYQ2DATA: is also valid, but must include the colon character, so it is distinct from an app-based name. The default extension when no extension is supplied is .dat.

app/filename.[extension

Is an app-based logical name of the data source. The default extension when no extension is supplied is .dat.

Example: Overriding DATASET and RMSFILE With a FILEDEF Command

You can override the DATASET value in the Master File or the RMSFILE declaration in the Access File with a FILEDEF command. The FILEDEF can be done locally within a procedure (before accessing the file) or within the profile. For example,

```
FILEDEF rmsfile DISK {'filename'|app/filename.[extension]}
```

where:

rmsfile

Is the logical metadata reference name matching the physical file name.

filename

Is the platform-dependent physical name of the data source. In a prior release, the physical name was required to be in single quotes. This is still valid syntax, but not required. A direct logical name such as MYQ2DATA: is also valid, but must include the colon character, so it is distinct from an app-based name. The default extension when no extension is supplied is .dat

app/filename.[extension

Is an app-based logical name of the data source. The default extension when no extension is supplied is .dat.

When to Use an Access File With an RMS Data Source

An Access File is optional for an RMS data source if the DATASET attribute is used in the Master File and if WRITE access is not required. However, even in those instances, you may wish to have an Access File to specify ACCESS, LOCKMODE, and STATMODE for RMS files:

- The ACCESS attribute can have a value of SHARED for read/write access, or READONLY for Read access. READONLY is the default setting when no explicit ACCESS parameter is present.
- The LOCKMODE attribute is primarily for retrieval (TABLE / SQL SELECT) only, but does have an effect on MODIFY / SQL UPDATE, so a site may want to consider separate Master Files for this purpose. LOCKMODE can have a value of REJECT, SKIP or KEEP. LOCKMODE not declared is the equivalent of REJECT.
- STATMODE attribute can have a value of EXCEPTIONS, ON, or OFF. EXCEPTIONS is the default and displays a message when locked records are encountered. ON allows a message to occur (whether or not locked records are encountered), and OFF suppresses locking messages.

See *File and Record Locking* on page 2081 for more information on ACCESS, LOCKMODE, and STATMODE. For related information, see *Handling Locked Records During Table Read Request* on page 2082.

File and Record Locking

This topic describes features that enable a developer to control file and record locking. Lock conflict, or contention, can occur at file level or record level. File contention is caused by incompatible access by two or more processes. Record contention is caused when a requested lock is incompatible with any existing locks on the record.

The following terms summarize the types of access allowed to RMS files and records:

Access Type	Description
Read Access	Provides users with access to a file and the records in the file for actions that make no alterations. Read can use any access mode (dependent on site needs) for the actions that are defined in the following tables.
Read/Write Access	Provides users with access to a file and the records in the file for actions that make alterations to records, add records, or delete records. Read/write access requires the use of SHARED as the access mode. All other modes do not support WRITE operations.

File Locking and the Interface to RMS

When your request attempts to open an RMS file, the Access Parameter within the Access File determines what you can do at a file level with the RMS file, and what actions others can take while you have the file open.

When a request attempts to open an RMS file, the Access Parameter determines which type of access to use. If the Access Parameter specification is incompatible with another process' access to the file, the procedure fails because it is not able to open the file. Additionally, if the file is opened for read, but a subsequent write is attempted, it will fail due to an inappropriate access mode.

Depending on how a file is opened, as specified by the Access Parameter, subsequent opens are limited as follows:

		Processes by Others			
Processes by Server Users	READONLY	Allowed	Allowed	Allowed	Denied
	SHARED	Allowed	Allowed	Allowed when SHAREd process is doing read-access operation.	Denied

This table describes user access and file sharing options used in response to the following Access File options for the Access Parameter.

Access Parameter	User Access	File Sharing to Other Processes
READONLY	Read	Read and Write
SHARED	Read and Write	Read and Write

Record Locking

RMS requests are no-lock for each record retrieved for READ operations during TABLE or SELECT. RMS operations for UPDATE and DELETE will take a lock on the record with SYS$GET for the duration of the operation and will check the return code status. An INSERT checks for existence, does SYS$PUT and checks the return code status.

Handling Locked Records During Table Read Request

RMS files can be accessed while they are simultaneously accessed by other programs.

For example, the file might be in use by another program that is maintaining the data by adding, deleting, or updating records.

A lock on a record affects the ability of the server to access a record depending on whether the request is for READ (TABLE or SELECT) or for an UPDATE or DELETE operation.

RMS files can be accessed while they are simultaneously accessed by other programs. For instance, the file might be in use by another program that is maintaining it by adding, deleting, or updating records.

When a READ operation is rejected due to a lock conflict by another process (and the Access File does not have LOCKMODE, or LOCKMODE is set to REJECT), the request is aborted and the following message is displayed:

```
(FOC1325) RMS READ LOCK ABORT ERROR
```

You can override this behavior on a file-by-file basis using the LOCKMODE and STATMODE options in the Access File. LOCKMODE and STATMODE do not apply to WRITE requests such as UPDATE or DELETE, but they do affect general behavior.

LOCKMODE

The LOCKMODE parameter controls server behavior when a locked record is encountered. LOCKMODE has three valid values (REJECT, KEEP, and SKIP, as described below), and respective behaviors. The default when LOCKMODE is not explicitly declared is REJECT.

LOCKMODE READ Behaviors.

- **LOCKMODE = REJECT (or Not Declared).** On READ, when a specific record read operation is rejected due to a lock conflict by another process, the whole request (SELECT or TABLE) is aborted and the following message is displayed:

  ```
  (FOC1325) RMS ABORT : %RMS-E-RLK, target record currently locked by
  another stream
  ```

  Records retrieved before the lock is encountered may have already been sent to the client (displayed) and should be discarded.

- **LOCKMODE = KEEP.** On READ, retrieves all records, including locked records and completes the request.

- **LOCKMODE = SKIP.** On READ, retrieves all records, except locked records and completes the request.

LOCKMODE UPDATE and DELETE Behaviors.

- **LOCKMODE = REJECT (or Not Declared).** On UPDATE or DELETE, when an operation is rejected due to a lock conflict by another process, the request is immediately aborted and the following message is displayed:

  ```
  (FOC1325) RMS ABORT : %RMS-E-RLK, target record currently locked by
  another stream
  (FOC1303) RMS RECORD NOT FOUND filename
  ```

- **LOCKMODE = KEEP.**

  - On UPDATE or DELETE, when a record is locked due to a lock conflict by another process, the request is waited with a delay cycle (10 seconds by default). If the record frees up during the period the record is processed, otherwise it is rejected as a timeout with the following message:

```
(FOC1301) RMS TIME OUT ERROR: {file}/%RMS-W-TMO, timeout period
expired
```

The delay period is 10 seconds and is adjustable (see *LOCKMODE=KEEP Wait Time* on page 2084). The delay is set at the RMS file open time and can not be adjusted on a record by record basis. If the locked record has actually changed since original retrieval the request will fail with:

```
(FOC526)TRANS     1 INVALID OTHER USER CHANGED filename
```

Failure recovery logic may be written into an application (that is, simple retry, recompute and retry, or continue) on the assumption that the record will be unlocked on the re-attempt. It is up to the application developer to write any such recovery logic, there are no automatic retries in this scenario.

- **LOCKMODE = SKIP.** On UPDATE or DELETE, when an operation is rejected due to a lock conflict by another process, the request is immediately rejected and the following message is displayed:

```
(FOC415)TRANS     1 REJECTED  NOMATCH filename
```

LOCKMODE does not affect INSERT operations because a lock cannot exist on a record that does not exist. The adapter will, appropriately, reject a record if another process inserts the same record before the adapter does, it is just not a LOCKMODE failure.

LOCKMODE=KEEP Wait Time

LOCKMODE=KEEP has a built-in 10 second delay upon encountering a record under UPDATE or DELETE modes. The purpose is to enable a developer to write recovery logic for UPDATE in case a record changed between when it was initially accessed and the actual adapter UPDATE attempt, but to not let the re-attempt happen so quickly that the record would still be locked.

The LOCKMODE=KEEP delay time is also settable using the following syntax.

```
ENGINE RMS SET WAIT_LOCKED n
```

where:

n

Is the wait time in number of seconds.

The delay can never be set to zero (use an alternate no-delay LOCKMODE value for zero). Specifying the value zero (0) results in the default value of 10.

STATMODE

STATMODE controls whether or not a message is sent to the client program about retrieved records. STATMODE has three settings:

- ON

  To receive a message giving the statistics of data retrieval, set STATMODE to ON in the Access File. The message displays, in addition to the report request, after data retrieval is complete. For example:

  ```
  STATMODE = ON
  ```

  A FOC1320 message displays when no locked records are encountered. For example:

  ```
  (FOC1320) RMS STATS  :(SEGCAR   ) Reads = 1, Skips = 0, Keeps = 0
  ```

  If LOCKMODE is set to KEEP a FOC1324 message displays. For example:

  ```
  (FOC1324) RMS KEEP   :(SEGCAR   ) Reads = 1, Skips = 2, Keeps = 0
  ```

  If LOCKMODE is set to SKIP, a FOC1322 message displays. For example:

  ```
  (FOC1322) RMS SKIP   :(SEGCAR   ) Reads = 1, Skips = 0, Keeps = 2
  ```

- OFF

  Set the STATMODE to OFF if no messages are to be sent when locked records are encountered. The client application will not have any indication whether or not locked records were encountered during record retrieval.

- EXCEPTIONS

  To receive a message giving the statistics of data retrieval, only if locked records are encountered, set the STATMODE to EXCEPTIONS. The message displays, in addition to the report request, after data retrieval is complete. No message displays if no locked records are encountered during retrieval. For example:

  ```
  STATMODE = EXCEPTIONS
  ```

  If LOCKMODE is set to KEEP a FOC1324 message displays. For example:

  ```
  (FOC1324) RMS KEEP   :(SEGCAR   ) Reads = 1, Skips = 2, Keeps = 0
  ```

  If LOCKMODE is set to SKIP, in addition to the report output, a FOC1322 message displays. For example:

  ```
  (FOC1322) RMS SKIP   :(SEGCAR   ) Reads = 1, Skips = 0, Keeps = 2
  ```

Example: Syntax Samples for an Access File

The following is an example of a Remote Node:

```
RMSFILE=HOST1::DISK$PROG:[PROD.INFO]EMPINFO.DAT,ACCESS=SHARED,$
```

The following is an example of a Full Path as defined by a logical and a file name:

```
RMSFILE=DATADIR:PERSON.DAT,ACCESS=READONLY,$
```

The following is an example of a Logical Name:

```
RMSFILE=MYLOGICAL:,ACCESS=SHARED,$
```

The following is an example of a Full Path on a file using LOCKMODE and STATMODE:

```
RMSFILE=DISK100:[DATA]PERSON.DAT,ACCESS=READONLY,LOCKMODE=SKIP,
  STATMODE=OFF,$
```

Retrieving Data From RMS Files

The following topics outline the procedures necessary for retrieving data from RMS files.

Index Selection

By default, the primary key is used for retrieval of records from indexed RMS files. You can override this default with the Automatic Index Selection (AIS).

The primary benefit of keys is improved efficiency in record retrieval. They provide an alternate, more efficient, retrieval method and can be used with screening tests on the selected key which are translated into direct reads. Indexes can also be used to control the order of retrieval of records.

To screen more than one field when working with group keys that are comprised of multiple fields, use a slash (/) character to separate the components of the group key. Screening is limited to NE, IS, and EQ relationships. The syntax is

```
... groupname {NE|IS|EQ}['']value/value.../value['']
```

where:

groupname

Is the GROUP field name in the Master File.

The slash (/) separates the parts of the group field. If you omit the optional quotation marks, embedded blanks in the value will be removed. Use single quotations marks (') around the values to preserve the values exactly as typed.

Automatic Index Selection (AIS)

The Automatic Index Selection (AIS) facility automatically selects a key for direct access to an indexed file based on a request.

Requirements for Using AIS

AIS automatically uses a key for direct retrieval when an applicable screening condition is reached in a request. AIS is used when all of the following requirements are met:

- Primary keys are described in a Master File with the ALIAS=KEY or ALIAS=DKEY attribute placed on the GROUP declaration (not on every field in the GROUP). Note that the primary key must always be described as a GROUP.
- Secondary keys are described on the FIELD declaration for a secondary key not belonging to a group.
- There is an optimizable screening condition on a key field. If a key is defined as a GROUP, the screening condition must be on either the GROUP name, or on the first field of the key. Only those key fields in a screening condition with one of the following logical relations are used for index selection:
  - Equal
  - Less than or equal to
  - Less than
  - Greater than
  - Greater than or equal to
- AIS applies only to keys in the root segment.

Syntax for RMS Master File Attributes

This topic details the syntax for the attributes used to describe files, segments, and fields.

File Attributes

File Declaration:

```
FILE[NAME]=filename, SUFFIX=type, {DATASET|DATA}='filename',$
```

Attribute	Function	Value
FILE[NAME]	Identifies the Master File.	All characters and digits, but no embedded blanks. Eight characters maximum.
SUFFIX	Identifies the type of file to which the Master File applies.	RMS.
DATASET|DATA	Identifies the physical location of the data source. Optional attribute. The RMSFILE attribute in the Access File or a FILEDEF command are alternatives. For more information, see *Associating an RMS Data Source to a Master File* on page 2078.	Is the platform-dependent physical name of the data source.

Note: The SUFFIX values listed here pertain to the RMS files. The SUFFIX parameter can have other values for data that resides on other Database Management Systems that can be accessed. Consult the appropriate Adapter manuals for these products.

Except for the following differences, the description of data and relationships between fields within a file is the same for keyed (indexed) and non-keyed RMS files, FIX (fixed-format sequential) files, and COM (comma-delimited) files. Keyed and non-keyed RMS file use RMS File System API calls to access data. The others use sequential file reads for accessing the data source. Index-related declarations do not apply to fixed-sequential or comma-delimited files.

Segment Attributes

Segment Declaration:

```
{SEGNAME|SEGMENT}=segname[,SEGTYPE=S0][,PARENT=parent_name]
            [          {n        }]
            [, OCCURS= {fieldname}]  [, POSITION=fieldname],$
            [          {VARIABLE }]
```

Attribute	Function	Value
SEGNAME SEGMENT	Identifies a collection of data fields that are related.	All characters and digits, except special characters. Up to eight characters in length.
SEGTYPE	Identifies the physical storage of the segment.	S0.
PARENT	Identifies the "parent" or owner of the current segment.	Any valid segment name previously defined in the Master File.
OCCURS	Identifies the field as a multiply-occurring field and specifies how the number of occurrences will be determined.	Any integer from 1 to 4095 or any valid field name or VARIABLE.
POSITION	Identifies the field in the parent that marks the beginning of the multiply-occurring fields.	All characters and digits, except special characters.

Field Attributes

Field Declarations:

```
                {fieldname}         {alias                }
FIELD[NAME]=    {FILLER    } ALIAS= {rectype identifier},USAGE=usage,
                {RECTYPE   }        {ORDER              }
                {MAPVALUE  }        {KEY(n)             }
                                    {MAPFIELD           }

ACTUAL=actual [,ACCEPT{list}]
[,TITLE='text'][,DESCRIPTION]=description],$

GROUP= {groupname|keyname} ,ALIAS={DKEY(n)|KEY(n)}
       ,USAGE=usage,ACTUAL=actual ,$

[DEFINE name/format=expression;$]
```

Attribute	Function	Value
FIELDNAME	Uniquely identifies a data item in the file.	All characters and digits, with a maximum of 64.
ALIAS	Alternate field name, sometimes used to provide additional information about the field.	All characters and digits, except special characters, with a maximum of 64.
USAGE	Describes the format of the field as interpreted for usage (display) purposes.	See *RMS Attribute Summary* on page 2090.
ACCEPT	Assigns a list or range of acceptable values for RECTYPE or MAPVALUE.	See *RMS Attribute Summary* on page 2090.
ACTUAL	Describes the format of the field as it exists in the external file.	See *RMS Attribute Summary* on page 2090.
TITLE	Provides one or more lines of text to be used as an alternate column heading for the field name on reports.	All characters and digits with a maximum of 64.
DESCRIPTION	Documents the meaning of a field in the Master File.	All characters and digits, with a maximum of 44.
GROUP	Identifies a key.	All characters and digits, except special characters, with a maximum of 48.
DEFINE	Creates a temporary field for reporting purposes.	Mathematical or logical statement made up of constants, field names, or other DEFINE fields.

RMS Attribute Summary

This topic contains a summary of the attributes.

Throughout the summary we refer to special characters and denote this reference with an "*". You should avoid using the special characters listed below, as they may impose operational restrictions in some environments.

+ (plus sign)	**'** (apostrophe)	**"** (double quotation marks)
- (hyphen)	**;** (semicolon)	**<** (less than sign)
$ (dollar sign)	**b/** (blank)	**>** (greater than sign)
* (asterisk)	**,** (comma)	**=** (equal sign)
/ (slash)	**|** (concatenation symbol)	**.** (period)
() (parentheses)		

Attribute	**ACCEPT**
Length:	1 to 255 characters
Example:	`ACCEPT = A OR B OR C`

Attribute	ACCEPT

Function:	Is optional, and enables you to assign a list or range of acceptable values to a RECTYPE or MAPVALUE field in a file.
	
	The syntax is
	
	`ACCEPT = {list|range}`
	
	where:
	
	list
	
	Is a string of acceptable values:
	
	value1 OR value2 OR value3...
	
	For example, ACCEPT=RED OR WHITE OR BLUE. You can also use a blank as an item separator. If the list of acceptable values runs longer than one line, continue it on the next line. The list is terminated by a comma (,).
	
	range
	
	Gives the range of acceptable values:
	
	value1 TO value2
	
	For example:
	
	`ACCEPT = 150 TO 1000`

Attribute:	ACTUAL
Length:	1 to 8 characters.
Function:	Describes the type and length of a field as it actually exists in a file. The source of this information is an existing description of the file. The following chart shows the data format types that the server reads:

Attribute:	ACTUAL	
	Type	**Meaning**
	An	Alphanumeric character string (A-Z, 0-9, and other ASCII display characters), where *n* = 1-4,095.
	D8	8-byte double-precision floating-point numbers.
	F4	4-byte single-precision floating-point numbers.
	Pn	Packed decimal internal format. The length is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign (+ or -). P6 means 11 digits plus a sign, packed 2 digits to the byte, for a total of 6 bytes of storage, where *n* = 1-8.

Attribute :	ACTUAL	
	In	Binary integers: I1 - Single-byte unsigned, binary integer (unsigned byte) I1.*n* - Single-byte signed, scaled binary integer (signed byte) I2 - 2-byte signed, binary integer (signed word) I2.*n* - 2-byte signed, scaled binary integer (signed word) I4 - 4-byte signed, binary integer (signed longword) I4.*n* - 4-byte signed, scaled binary integer (signed longword) I8 - 8-byte signed, binary integer (signed quadword) I8.*n* - 8-byte signed, scaled binary integer (signed quadword) If an integer field contains an assumed decimal point (meaning it is a scaled integer), the field is represented as I*m*.*n*, where *m* is the total number of storage bytes, and *n* is the number of decimal places for the scaling. For example, I4.1 means a 4-byte number with one decimal place. If I1 (unsigned) is used with negative data values, incorrect values will display. It is important to know if the data will contain negative values and indicate that it is signed by using a scale factor, even if it is simply I1.0 (to indicate single byte signed, no scaling). This rule for signed and unsigned is only unique to ACTUAL I1 and does not apply to I2, I4 or I8 sizes USAGE on a scaled integer may be P, D or I, but should be minimally configured as a utype*m*+2.*n* (for example, I12.2) to account for not having overflows on negative values. However, a USAGE of I is limited to an *m* value of 11. It should also be noted that JDBC JLINK access to a Master File using scaled integers must use USAGEs of P or D for scaling because scaled I is not supported in the JDBC specification. Octaword Integer data type (USAGE=I16) is not supported.

Attribute:	ACTUAL	
	Z*n*	Zoned decimal format (numeric string) where *n* is the number of digits (1 to 31), each of which takes one byte of storage. The last byte contains a digit and the sign. There are several standards for zoned data. For Read purposes, only right overpunched standards are supported and can be determined on Read since they are unique. The specific format to use when writing data must be known for read/write purposes. The default is ASCII right overpunch. To change the default, you must edit the EDACONF [.BIN]EDAENV.COM file and add the following logical: `DEFINE /NOLOG IBI_ZONED_OUT_TYPE {1|2}` where: 1 Is the ASCII right overpunched standard (default). 2 Is the EBCDIC right overpunched standard. Zoned right separate numeric or zoned left overpunched numeric formats are not supported.

<table>
<tr><th>Attribute :</th><th>ACTUAL</th><th></th></tr>
<tr><td></td><td>DATE</td><td>Unless your file was created by a program, all of the characters will be in ASCII format type A (alphanumeric).

The server permits the following conversions from ACTUAL format to USAGE (display) format:<table><tr><th>USAGE</th><th>ACTUAL</th></tr><tr><td>I</td><td>I</td></tr><tr><td>P</td><td>P</td></tr><tr><td>P,D,I,F</td><td>Z</td></tr><tr><td>D</td><td>D</td></tr><tr><td>F</td><td>F</td></tr><tr><td>P</td><td>I8</td></tr><tr><td>YYMD</td><td>DATE</td></tr></table></td></tr>
<tr><td></td><td>H</td><td>Date-Time stamp data that uses formatting options to display raw data or formatted time data, date data, or date-time data. See the Describing Data With WebFOCUS Language manual for a full list of options. By default, the values used in this feature are the FOCUS internal time keeping system, but the setting:

SET VMSTIMESTAMP = VMS

Lets actual data based on the OpenVMS Native 64-bit DEC Date specification be read and handled seamlessly.

Note: Releases prior to 7.7.07 required declaration of DEC Date as A8, and the use of the HINPUT()/CVTSTIMES() functions, which was less than seamless. It is recommended that any such metadata be converted to use H dates and this setting.</td></tr>
</table>

Attribute:	ALIAS
Alias:	SYNONYM
Length:	1 to 12 characters
Permitted Values:	All characters and digits.
Example:	`ALIAS=CTY`
Function:	Is an alternate name to identify a data field. Since references to data fields are based on the names, aliases, or shortest unique truncation, it is useful to make the ALIAS a short abbreviation representative of the data. Short, simple names are best, with no embedded blank spaces or special characters. For example, PART CODE or PART-CODE should be PARTCODE or PART_CODE, or PC. Names must begin with a letter, but can contain numbers. Do not use names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...) or HOLD file aliases (E01, E02, ...). Also, avoid reserved keywords such as BY or ACROSS. ALIAS has specialized functions when used with KEY definitions, RECTYPE and MAPFIELD, as discussed at the beginning of this manual.

Attribute:	DEFINE
Example:	`DEFINE PROFIT/D8 = RETAIL_COST - DEALER_COST;$`

Attribute:	DEFINE
Function:	Is optional, and enables you to store definitions for temporary fields in your Master File for reporting purposes. The syntax is `DEFINE name[/format] = expression;$` where: *name* Is a 1 to 48 character field name for the defined field. *format* Is the optional display format for the defined field, separated from *name* by a slash (/). The display format for the field follows the rules for USAGE formats described under field attributes. The default value is D12.2. *expression* Can be either a mathematical or logical statement, made up of constants, database fields, and temporary defined fields. The expression must end with a semicolon (;) followed by a dollar sign ($). A DEFINE is placed at the end of the segment it is associated with. A defined field stored in a Master File can refer only to fields in its same segment. To create a defined field that refers to fields in other segments, create a temporary field using the DEFINE command prior to your report request.

Attribute:	DESCRIPTION
Length:	1 to 44 characters
Permitted Values:	All characters and digits.
Example:	`DESCRIPTION=STANDARD COST CATEGORY,$`

Attribute:	DESCRIPTION

Function:	Is optional, and is used only for documenting the meaning of a field in the Master File. It is ignored during processing. Since a Master File is a comma-delimited file that the server can read directly, it is possible to prepare reports in various formats whose subjects are the names and attributes of the data fields.
	
	If the DESCRIPTION contains a comma (,), the entire text must be enclosed in single quotation marks ('). For example:
	
	`DESCRIPTION='TOTAL COST, NOT NORMALIZED',$`
	
	Another way to add comments to a Master File is to place them after the dollar sign ($).
	Descriptions of DEFINE fields in the Master File are placed on separate lines. For example:
	
	`DEFINE ITEMS_SOLD/D8 = INVENTORY - ORDERED`
	`;DESC=DAMAGED ITEMS NOT INCLUDED,$`
	
	Note: When used on a DEFINE expression, the dollar sign ($) does not immediately follow the semicolon (;).
	
	The semicolon (;) after a DEFINE must display on the same line as the attribute, which follows the DEFINE.

Attribute:	FIELDNAME
Alias:	FIELD
Length:	1 to 64 characters
Permitted Values:	All characters and digits.
Example:	`FIELD=INVENTORY`

Attribute:	FIELDNAME
Function:	Identifies the data items in a file. Names must be unique within a file. The full field name is used as the default title for the data printed on reports. Hence, names representative of the data should be selected. Names must begin with a letter, though they may contain numbers. Avoid names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...) or HOLD file aliases (E01, E02, ...). Also avoid reserved keywords such as PRINT, BY, ACROSS, and so on. FIELDNAME has specialized functions when used with RECTYPE and MAPVALUE, as discussed at the beginning of this manual.

Attribute:	FILENAME
Alias:	FILE
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.
Example:	`FILENAME=EQUIP`
Function:	Is optional, and is used for documentation purposes. It should correspond to the external file name of the Master File.

Attribute:	GROUP
Alias:	KEY
Length:	1 to 66 characters
Permitted Values:	All characters and digits, but no embedded blanks.
Example:	`GROUP=PUBKEY`

Attribute:	GROUP
Function:	Is used to describe the primary key in a keyed file or a field that is composed of subfields. The USAGE format and ACTUAL format of the GROUP is calculated using the USAGE and ACTUAL formats of the FIELDS that comprise the GROUP. Names must begin with a letter, though they may contain numbers. Avoid using names that might conflict with report column names (C1, C2, ...), row names (R1, R2, ...), or HOLD file aliases (E01, E02, ...).

Attribute:	PARENT
Alias:	PARENT
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.
Example:	`PARENT=CARSEG`
Function:	Is the name of the segment that is the parent or "owner" of the current segment. In the Master File, the information describing the parent must precede any reference to it as a parent. The first, or "root" segment, in a file cannot have a parent by definition, but the name "SYSTEM" may be used, or the attribute left blank. Every other segment must have a parent named. If no parent is named, the immediately preceding segment will be used by default.

Attribute:	SEGNAME
Alias:	SEGMENT
Length:	1 to 8 characters
Permitted Values:	All characters and digits, but no embedded blanks.

Attribute:	SEGNAME
Example:	SEGNAME=MODSEG
Function:	Identifies a collection of data fields. Segments which have a relationship to each other must have unique names within a given file description.

Attribute:	SUFFIX	
Alias:	FILESUFFIX	
Length:	1 to 8 characters	
Permitted Values:	RMS	
Example:	SUFFIX=RMS	
Function:	In order to identify the type of file the description applies to, the SUFFIX is given one of the following values:	
	Value	**Meaning**
	RMS	A keyed RMS file, which uses a Master File DATASET= attribute and an Access File RMSFILE= attribute or a FILEDEF to locate and access data using the RMS System and its Indexes.
	ISAM	An alternate suffix value for a keyed RMS file. ISAM is supported as an alternate to the RMS keyword for backward compatibility to FOCUS 6.x applications that used ISAM as its keyword.

Note: SUFFIX values for other proprietary databases that can be accessed are described in the appropriate manuals for these options.

Attribute:	TITLE
Length:	1 to 64 characters.

Attribute:	TITLE
Permitted Values:	All characters and values.
Example:	`TITLE='Products'`
Function:	Is optional, and enables you to supply an alternate column title to replace the field name that is normally used. The syntax is `TITLE='text'` where: *text* Must be enclosed within single quotation marks (') and can contain 1 to 64 characters. TITLE cannot span more than one line in the Master File. If necessary, move the entire attribute to a line by itself. To display the TITLE on more than one line in the report, use commas to divide the text. To include blanks at the end of a column title, simply type them in and enter a slash (/) in the final blank position. TITLE attributes do not apply when direct operations (PCT., AVE., ...) are used on the field. To rename such columns, use the AS phrase.
Changes:	You can override both field names and TITLE attributes with AS phrases in your request. You can issue a SET TITLE command to change the default titles to either the field names or titles supplied in the Master File. You can change the TITLE attribute in the Master File. **Note:** Client tools using the API do not have access to the TITLE attribute. The TITLE attribute is available only with WebFOCUS.

Attribute:	USAGE
Length:	1 to 8 characters

Attribute:	USAGE
Function:	Defines the report display format of the data field. This attribute defines the data field type, length, and any edit options that are to be applied when the field values are printed. Special date formats, which are actually an extended series of edit options, can also be used. The syntax is USAGE=*usage* where: *usage* Consists of three parts: *tllleeeee* *t* = field type *lll* = field display length *eeeee* = edit options

Permissible USAGE field types and lengths are shown in the chart below:

USAGE	Length	Description
A	1-4,095	Alphanumeric text
D	1-19	Decimal, double-precision numbers
F	1-9	Decimal, single-precision numbers
I	1-11	Integer values (no decimal places)
P	1-17	Packed decimal numbers
YYMD	10	Displayed as YYMD
H		Date-Time stamp data that uses formatting options to display raw data or formatted time data, date data or date-time data. See the *Describing Data With WebFOCUS Language* manual for a full list of options

Edit options only affect printed or displayed fields. They are not active for extract files.

The following table summarizes the edit options and includes sample USAGE values:

Edit Option	Meaning	Effect
%	Percent sign	Percent sign Displays a percent sign along with numeric data. Does not calculate the percent.
B	Bracket negative	Encloses negative numbers in parentheses.
c	Comma suppress	Suppresses the display of commas. Used with numeric format options M and N (floating and non-floating dollar sign) and data format D (floating-point double-precision).
C	Comma edit	Inserts a comma after every third significant digit, or a period instead of a comma if continental decimal notation is in use.
DMY	Day-Month-Year	Displays alphanumeric or integer data as a date in the form day/month/year.
E	Scientific notation	Scientific notation Displays only significant digits.
L	Leading zeroes	Adds leading zeroes.
M	Floating $ (for US code page)	Places a floating dollar sign $ to the left of the highest significant digit. **Note:** The currency symbol displayed depends on the code page used.
MDY	Month-Day-Year	Displays alphanumeric or integer data as a date in the form month/day/year.
N	Fixed $(for US code page)	Places a dollar sign $ to the left of the field. The symbol displays only on the first detail line of each page. **Note:** The currency symbol displayed depends on the code page used.

Edit Option	Meaning	Effect
R	Credit (CR) negative	Places CR after negative numbers.
S	Zero suppress	Zero suppress If the data value is zero, prints a blank in its place.

Read/Write Usage Limitations of the Adapter for RMS

The adapter has certain limitations when used for write purposes. The following topics discuss these issues and possible alternatives that may be used.

OCCURS Statements in a Master File

OCCURS statements in a Master File are not directly supported by the write portion of the adapter. If an OCCURS statement is for a specific number of instances, an alternate Master File may be coded by using specific naming (for example, AMTDUE OCCURS=12 would be coded as AMTDUE01, AMTDUE02, and so on). The write application must then reference the alternate Master File using the specific fields.

For example:

```
SEGNAME=MTH, SEGTYPE=S0, PARENT=YEAR, OCCURS=12,$
FIELD=PAYABLES, AP, I8 , I4 , $
```

would be coded as

```
SEGNAME=MTH, SEGTYPE=S0, PARENT=YEAR, $
FIELD=PAYABLES01, AP01, I8 , I4 , $
FIELD=PAYABLES02, AP02, I8 , I4 , $
FIELD=PAYABLES03, AP03, I8 , I4 , $
FIELD=PAYABLES04, AP04, I8 , I4 , $
FIELD=PAYABLES05, AP05, I8 , I4 , $
FIELD=PAYABLES06, AP06, I8 , I4 , $
FIELD=PAYABLES07, AP07, I8 , I4 , $
FIELD=PAYABLES08, AP08, I8 , I4 , $
FIELD=PAYABLES09, AP09, I8 , I4 , $
FIELD=PAYABLES10, AP10, I8 , I4 , $
FIELD=PAYABLES11, AP11, I8 , I4 , $
FIELD=PAYABLES12, AP12, I8 , I4 , $
```

SQL Commit Processing

The Adapter for RMS, in conjunction with RMS itself, treats a single SQL statement as a complete unit of work, by default. This means that the server automatically commits after every SQL statement. If the front-end application sends an SQL COMMIT or SQL ROLLBACK statement(s), it will be ignored by the adapter.

SQL Commands

The following are acceptable SQL commands:

Command	Function
`SELECT`	Retrieves data for the entire table (*) or for specified columns.
`DELETE`	Removes one or more records from an RMS keyed file.
`INSERT INTO`	Adds data to an RMS keyed file.
`UPDATE`	Updates values of one or more columns in a record of an RMS keyed file.

SQL, MODIFY and MAINTAIN Operations

Releases prior to 5.2 of the server did not support multi-record SQL INSERT, DELETE and UPDATE except when done as a PREPARE, plus MODIFY and MAINTAIN were not supported. This limitation is removed as of the 5.2 release. Recoding existing applications that used PREPARE is not required, all methods are supported.

Chapter 88

Using the Adapter for Rserve

The Adapter for Rserve enables you to define a connection to Rserve for remotely executing R scripts to be used with WebFOCUS reports and charts as a summary (Compute) or virtual field.

The original Rserve paper is available at *http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/Urbanek.pdf*.

For additional information on Rserve, see *https://rforge.net/Rserve/index.html*.

Note: If you encounter any errors when using the Adapter for Rserve, consult the Session Log for details.

In this chapter:

Introduction to the Adapter for Rserve

Rserve is a TCP/IP server that allows users to run R scripts directly from WebFOCUS or the Reporting Server without the need to initialize R or link to an R library.

The adapter is available on UNIX, Linux, and Windows. On UNIX and Linux, Rserve allows parallel sessions and, therefore, the environment supports multiple users. On Windows, Rserve does not support parallel connections. Multiple sessions are shared, so the environment supports a single user at a time.

The installation of R and Rserve, and the creation of the model are done without server or WebFOCUS involvement, and are prerequisites for configuring and using the Adapter for Rserve.

Prerequisites

1. Install R and Rserve on a host computer that has access to the WebFOCUS Reporting Server. Install them in a location for which you have write access.
2. Create an R model that you will want to run from WebFOCUS.

3. Save the R model (known as serialization) in a user script folder accessible to Rserve, using the R save command or the saveRDS function. The name and location of this folder will be determined by the Rserve administrator.
4. Create a .csv file that has only those columns that will be used as independent variables in the model. Do not include the dependent variable.
5. Write an R script to read and run the model.

R Script Features

The R script that will read and run the model must conform to the following rules:

- The R script must read command line arguments, even though a command line interface is not used. In the following example, the command line arguments are saved to a variable named args.

  ```
  args <- commandArgs(trailingOnly=TRUE)
  ```

- The first argument is the name of the file to be processed by the model. The file name and a temporary HOLD file are automatically generated by the server. The second argument is the name of the output file. For example:

  ```
  input_file <- file.path(args[1])
  output_file <- file.path(args[2])
  ```

- The result field name must match the output of the R script. For example, in the following syntax, the result field from the model is named Price:

  ```
  colnames(results) <- c('Price')
  write.csv(results, file=output_file, row.names=FALSE)
  ```

For a complete example, see *Sample Session: Creating a Synonym for an R Script and Running the Script* on page 2115.

Server and WebFOCUS Steps

Once you have a model and the R script that runs it, you need a sample .csv file that contains a few rows of data for the independent variables used in the model. This file will be used to determine the data types and lengths of the independent variables when you create a synonym for the model.

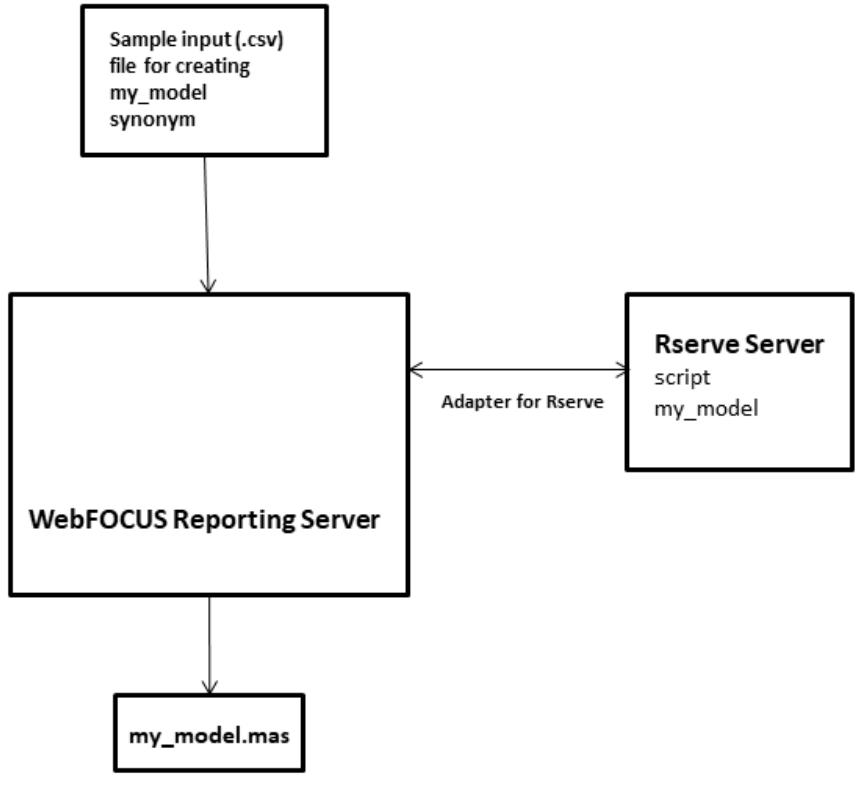
Later, when you run the model from WebFOCUS, you will need a data file to run against that will provide values for the input variables to be passed to the RSERVE function. The function will return the result of running the model as output.

Therefore, the following steps must be completed in the server Web Console and WebFOCUS.

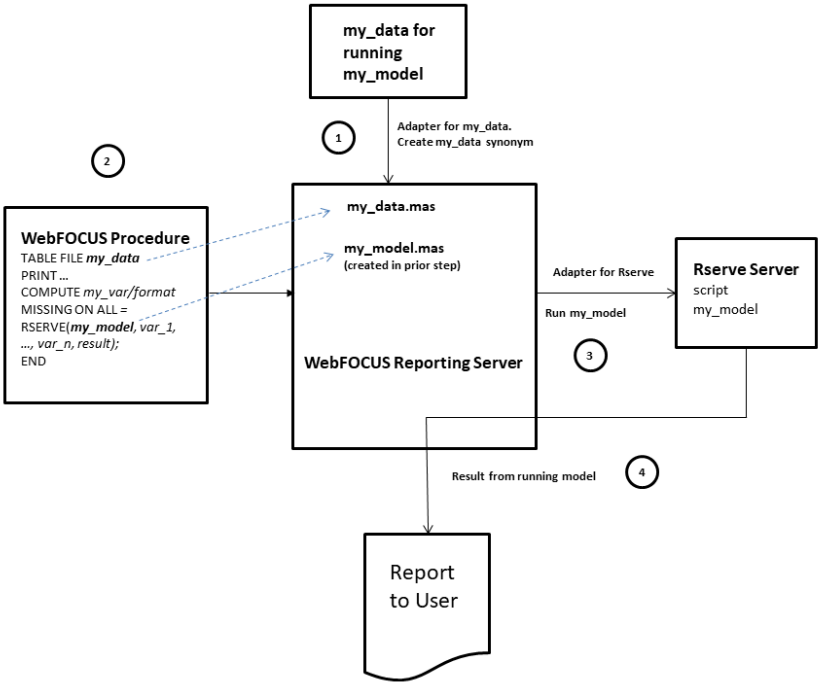
1. Configure the Adapter for Rserve, as described in *Configuring the Adapter for Rserve* on page 2112.

2. Upload the sample input file (.csv) to the server. This file is needed for creating the synonym for the model.
3. Create a synonym for the model, as described in *Creating a Synonym for an R Script* on page 2114.
4. Make sure an adapter has been configured for access to the data source against which you want to run the model.
5. Create a synonym for this data using the adapter configured in Step 4.
6. From WebFOCUS, create a report or chart that uses the RSERVE function in a COMPUTE, passes it the input values (independent variables), and retrieves the result, as described in *Sample Session: Creating a Synonym for an R Script and Running the Script* on page 2115.

The following diagram summarizes the process of preparing the model, named my_model, to be called in a WebFOCUS procedure.



The following diagram summarizes the process of calling the model named my_model in a WebFOCUS procedure (against a data source supported by R).



Configuring the Adapter for Rserve

For information about prerequisites for running the adapter, right-click the adapter name on the Available list or a connection for the adapter on the Configured Adapters list, and click *Prerequisites*.

To configure the adapter, on the *Connect to Data* page of the Reporting Server Web Console, right-click *Rserve* on the Available list, and click *Configure*.

The Add Rserve to Configuration page opens.

Enter the following connection parameters.

Connection Name
Is a name for this connection.

Server
Specifies the Rserve host machine.

When specifying a host machine name other than localhost, ensure that remote access is enabled in the Rserve configuration file by adding the following:

```
remote enable
```

For information on creating the Rserve configuration file, refer to the Rserve documentation.

Rserve does not have to be installed on the Reporting Server host machine. However, it must be accessible to the Reporting Server.

Port

Is the port on which the Rserve host listens.

Security

There are three methods by which a user can be authenticated when connecting to Rserve:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to Rserve, at connection time, for authentication.

  If you select Explicit authentication, enter the Rserve user ID and password.

- **Password Passthru.** The user ID and password received from the client application are passed to Rserve, at connection time, for authentication.

- **Trusted.** The adapter connects to Rserve using the rules for an impersonated process that are relevant to the current operating system.

IBI_CLASSPATH

Are additional Java Class directories or full path jar names to be available for Java Services, each on a separate line. You must add the path to the location of the following jar files to IBI_CLASSPATH.

- On UNIX and Linux, the files REngine.jar and RServeEngine.jar are required. They are available on the Rserve web site, *https://www.rforge.net/Rserve/files/*.

- On Windows, the files REngine.jar and Rserve.jar are required. They are available in the Rserve zip file you can install from the RGui or download from the Rserve web site, *https://www.rforge.net/Rserve/files/*.

Click *Configure*.

If the configuration is successful, the following message displays.

```
Rserve successfully added to configuration
```

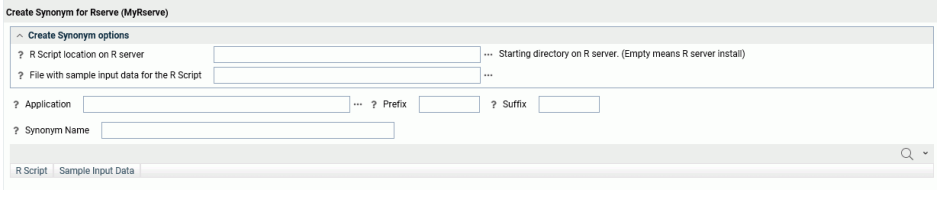
In addition, you can click *Test* on the context menu for the connection.

Creating a Synonym for an R Script

Each R script used with the Adapter for Rserve must have a synonym that describes the independent variables and dependent variable of the script. The Master File will contain the list of input (independent) variables and the output (dependent) variable. The Access File will contain information about the script and data files.

The synonym will be created using a sample file that contains only the fields that are input parameters for the script. A few rows of sample data are sufficient for the Adapter for Rserve to determine the appropriate data types and lengths of the parameters. The sample file must be a .csv file.

To create a synonym for an R script, right-click a connection for the Adapter for Rserve and click *Create metadata objects*. The Create Synonym for Rserve page opens, as shown in the following image, where the connection name is MyRserve.



Select or enter values for the following parameters.

R Script location on R server
Is the remote location, including script name and extension, for the R script file (.R). Leave blank if the R script is in an application folder accessible to the Reporting Server.

Note: In order to change a previously selected R script, you must delete the name of the file or the entire path before clicking the browse button.

Select file with sample input data for the R Script
Open the file picker (...) to select the application directory and file that contains the sample data for creating the synonym. Click *OK*.

Application
Open the file picker (...) to select the application that contains the R script. Select the R script file from the file picker and click *OK*.

Synonym Name
Enter a name for the resulting synonym, or accept the default name.

When you have finished entering the synonym creation parameters, click *Create Synonym* on the ribbon, as shown in the following image.

When you use the RSERVE function to run an R script from a WebFOCUS report or chart, the name and application location of this synonym will be passed as the first argument to the function. The other arguments will be the names of the independent variables and the dependent variable. For more information, see the *Using Functions* manual.

Sample Session: Creating a Synonym for an R Script and Running the Script

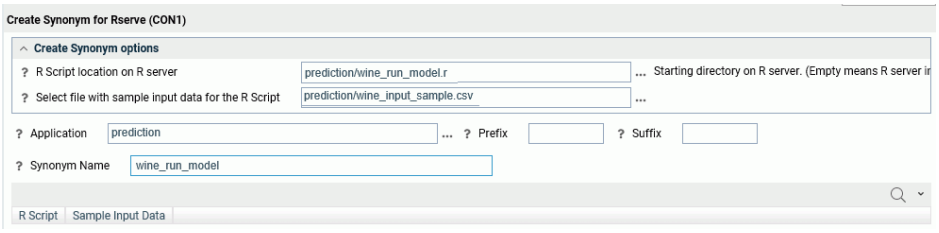
The following R script named wine_run_model.R predicts Bordeaux wine prices based on the average growing season temperature, the amount of rain during the harvest season, the amount of rain during the winter, and the age of the wine.

```
# filename: wine_run_model.r

args <- commandArgs(trailingOnly=TRUE)
input_file <- file.path(args[1])
output_file <- file.path(args[2])

wine_test <- read.csv(input_file)

wine_model <- readRDS('/prediction/wine_model.rds')

results <- predict(wine_model, newdata = wine_test)

results <- as.data.frame(results)
colnames(results) <- c('Price')
write.csv(results, file=output_file, row.names=FALSE)
```

The following sample data file named wine_input_sample.csv contains the names and sample values for the independent variables used in this model.

```
"AGST","HarvestRain","WinterRain","Age"
16.1667,122,717,4
16,74,578,3
```

The synonym creation page for this script is shown in the following image.



Clicking *Create Synonym* on the ribbon generates the wine_run_model synonym. The Master File, wine_run_model.mas, describes the independent (input) variables and the dependent (output) variable, as shown below:

```
FILENAME=WINE_RUN_MODEL, SUFFIX=RSERVE  , $
  SEGMENT=INPUT_DATA, SEGTYPE=S0, $
    FIELDNAME=AGST, ALIAS=AGST, USAGE=D9.4, ACTUAL=STRING,
      MISSING=ON,
      TITLE='AGST', $
    FIELDNAME=HARVESTRAIN, ALIAS=HarvestRain, USAGE=I11, ACTUAL=STRING,
      MISSING=ON,
      TITLE='HarvestRain', $
    FIELDNAME=WINTERRAIN, ALIAS=WinterRain, USAGE=I11, ACTUAL=STRING,
      MISSING=ON,
      TITLE='WinterRain', $
    FIELDNAME=AGE, ALIAS=Age, USAGE=I11, ACTUAL=STRING,
      MISSING=ON,
      TITLE='Age', $
  SEGMENT=OUTPUT_DATA, SEGTYPE=U, PARENT=INPUT_DATA, $
    FIELDNAME=PRICE, ALIAS=Price, USAGE=D18.14, ACTUAL=STRING,
      MISSING=ON,
      TITLE='Price', $
```

The Access File, wine_run_model.acx, describes the names and locations of the R script and the sample data file, as shown below.

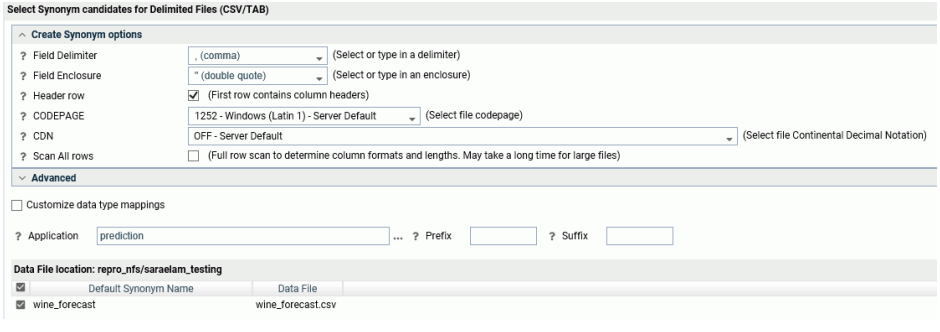
```
SEGNAME=INPUT_DATA,
  CONNECTION=MyRserve,
  R_SCRIPT=/prediction/wine_run_model.r,
  R_SCRIPT_LOCATION=WFRS,
  R_INPUT_SAMPLE_DAT=prediction/wine_input_sample.csv, $
```

Now that the synonym has been created for the model, the model will be used to run against the following data file named wine_forecast.csv.

```
Year,Price,WinterRain,AGST,HarvestRain,Age,FrancePop
1952,7.495,600,17.1167,160,31,43183.569
1953,8.0393,690,16.7333,80,30,43495.03
1955,7.6858,502,17.15,130,28,44217.857
1957,6.9845,420,16.1333,110,26,45152.252
1958,6.7772,582,16.4167,187,25,45653.805
1959,8.0757,485,17.4833,187,24,46128.638
1960,6.5188,763,16.4167,290,23,46583.995
1961,8.4937,830,17.3333,38,22,47128.005
1962,7.388,697,16.3,52,21,48088.673
1963,6.7127,608,15.7167,155,20,48798.99
1964,7.3094,402,17.2667,96,19,49356.943
1965,6.2518,602,15.3667,267,18,49801.821
1966,7.7443,819,16.5333,86,17,50254.966
1967,6.8398,714,16.2333,118,16,50650.406
1968,6.2435,610,16.2,292,15,51034.413
1969,6.3459,575,16.55,244,14,51470.276
1970,7.5883,622,16.6667,89,13,51918.389
1971,7.1934,551,16.7667,112,12,52431.647
1972,6.2049,536,14.9833,158,11,52894.183
1973,6.6367,376,17.0667,123,10,53332.805
1974,6.2941,574,16.3,184,9,53689.61
1975,7.292,572,16.95,171,8,53955.042
1976,7.1211,418,17.65,247,7,54159.049
1977,6.2587,821,15.5833,87,6,54378.362
1978,7.186,763,15.8167,51,5,54602.193
```

The data file can be any type of file that R can read. In this case it is another .csv file. This file needs a synonym in order to be used in a report request.

The following image shows the synonym creation page for wine_forecast.csv using the Adapter for Delimited Files.



The following is the generated Master File, wine_forecast.mas.

```
FILENAME=WINE_FORECAST, SUFFIX=DFIX    , CODEPAGE=1252,
   DATASET=prediction/wine_forecast.csv, $
SEGMENT=WINE_FORECAST, SEGTYPE=S0, $
    FIELDNAME=YEAR1, ALIAS=Year, USAGE=I6, ACTUAL=A5V,
      MISSING=ON,      TITLE='Year', $
    FIELDNAME=PRICE, ALIAS=Price, USAGE=D8.4, ACTUAL=A7V,
      MISSING=ON,      TITLE='Price', $
    FIELDNAME=WINTERRAIN, ALIAS=WinterRain, USAGE=I5, ACTUAL=A3V,
      MISSING=ON,      TITLE='WinterRain', $
    FIELDNAME=AGST, ALIAS=AGST, USAGE=D9.4, ACTUAL=A8V,
      MISSING=ON,      TITLE='AGST', $
    FIELDNAME=HARVESTRAIN, ALIAS=HarvestRain, USAGE=I5, ACTUAL=A3V,
      MISSING=ON,      TITLE='HarvestRain', $
    FIELDNAME=AGE, ALIAS=Age, USAGE=I4, ACTUAL=A2V,      MISSING=ON,
TITLE='Age', $
    FIELDNAME=FRANCEPOP, ALIAS=FrancePop, USAGE=D11.3, ACTUAL=A11V,
      MISSING=ON,      TITLE='FrancePop', $
```

The following is the generated Access File, wine_forecast.acx.

```
SEGNAME=WINE_FORECAST,   DELIMITER=',',   ENCLOSURE='',   HEADER=YES,
CDN=COMMAS_DOT,   CONNECTION=<local>, $
```

The following request, wine_forecast_price_report.fex, uses the RSERVE bulit-in function to run the script and return a report.

```
-*wine_forecast_price_report.fex
TABLE FILE PREDICTION/WINE_FORECAST
PRINT
  YEAR
  WINTERRAIN
  AGST
  HARVESTRAIN
  AGE

  COMPUTE PREDICTED_PRICE/D18.2 MISSING ON ALL=
    RSERVE(prediction/wine_run_model, AGST, HARVESTRAIN, WINTERRAIN, AGE, Price);
        AS 'Predicted,Price'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

The output is shown in the following image.

Year	WinterRain	AGST	HarvestRain	Age	Predicted Price
1952	600	17.1167	160	31	7.72
1953	690	16.7333	80	30	7.87
1955	502	17.1500	130	28	7.68
1957	420	16.1333	110	26	7.00
1958	582	16.4167	187	25	7.02
1959	485	17.4833	187	24	7.54
1960	763	16.4167	290	23	6.76
1961	830	17.3333	38	22	8.36
1962	697	16.3000	52	21	7.51
1963	608	15.7167	155	20	6.63
1964	402	17.2667	96	19	7.56
1965	602	15.3667	267	18	5.92
1966	819	16.5333	86	17	7.56
1967	714	16.2333	118	16	7.11
1968	610	16.2000	292	15	6.26
1969	575	16.5500	244	14	6.60
1970	622	16.6667	89	13	7.32
1971	551	16.7667	112	12	7.19
1972	536	14.9833	158	11	5.88
1973	376	17.0667	123	10	7.09
1974	574	16.3000	184	9	6.57
1975	572	16.9500	171	8	6.99
1976	418	17.6500	247	7	6.92
1977	821	15.5833	87	6	6.71
1978	763	15.8167	51	5	6.91

Chapter 89

Using the Adapter for Salesforce.com

The Adapter for Salesforce.com provides query access to salesforce objects using SOQL, the Salesforce Object Query Language.

In this chapter:

Configuring the Adapter for Salesforce.com

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Salesforce.com Connection Parameters

The following list describes the information in the Add Salesforce.com to Configuration dialog box. Not all options appear for all applications.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

User

Salesforce.com user ID (email address).

Password

The password associated with your Salesforce.com user ID, followed by the security token. For example, if your password is myPass and your security token is xyyzyx then you would enter myPassxyyzyz.

SALESFORCE URL

The URL to login to Salesforce.com using SOAP. Use the default value provided unless instructed otherwise by customer support.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

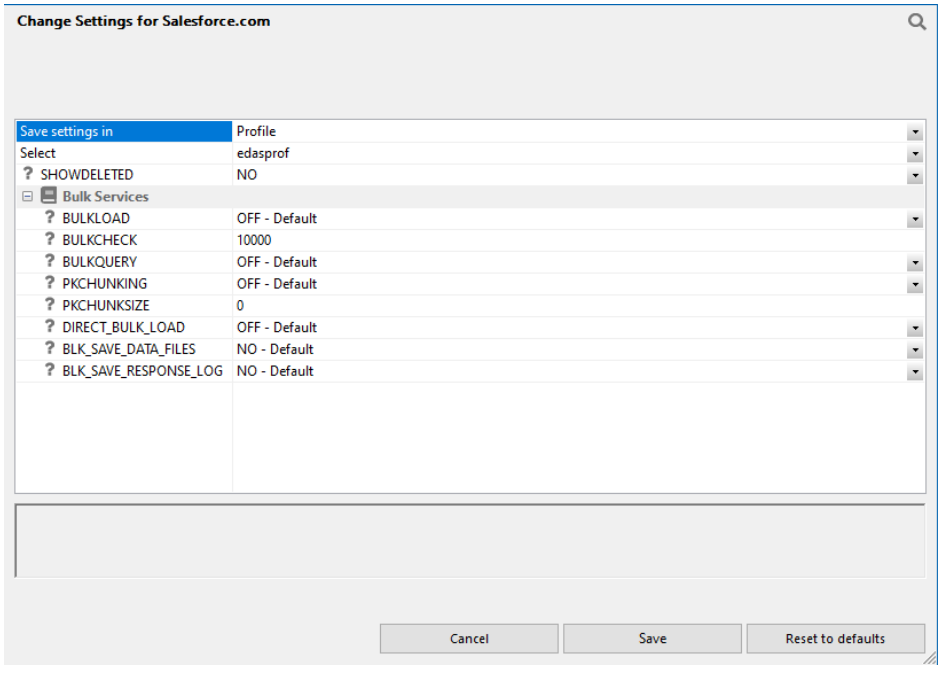
Store the connection attributes in the server profile (edasprof).

Bulk Retrieval With Salesforce.com

When working with Salesforce.com, there are various settings that you can set that will allow you to query large data sets and reduce the number of API requests.

These settings can be accessed from the Change Settings dialog box for Salesforce.com. To open this dialog box, from the DMC, expand the *Adapters* folder, right-click *Salesforce.com*, and click *Change Settings*.

The Change Settings dialog box opens, as shown in the following image.



The following Bulk Services options are available:

BULKLOAD
Enables the use of Extended Bulk Load. The default value is OFF.

BULKCHECK
The row interval to commit or write transactions to the database. The default value is 10000.

BULKQUERY
Enables Bulk Query for data retrieval. The default value is OFF. When this option is selected, queries that meet the Salesforce.com requirements, such as no aggregation, will use bulk query.

You can also use the following command to enable or disable Bulk Query:

```
ENGINE SFDC SET BULKQUERY {ON|_OFF_}
```

PKCHUNKING

Enables bulk query Primary Key chunking for data retrieval. The default value is OFF. When this option is enabled, extremely large datasets can be extracted. The Primary Key of the object is used to retrieve data in chunks, which allows for retrieval of larger data values.

You can also use the following command to extract extremely large datasets:

```
ENGINE SFDC SET PKCHUNKING {ON|_OFF_}
```

PKCHUNKSIZE

Enables the bulk query Primary Key chunk size for data retrieval. The default value is 0. When this value is not set, or set to 0, Salesforce.com will use the default chunk size of 100000. The maximum chunk size is 250000.

You can also use the following command to enable the bulk query chunk size:

```
ENGINE SFDC SET PKCHUNKSIZE {0..250000}
```

BLK_SAVE_DATA_FILES

When set to *Yes*, retains an intermediate file that is created, as well as the log (response) file. The default value is *No*.

The files are saved in the same application directory as the flow. Each is generated with a synonym that you can use for reporting. The file type of the intermediate HTML file is .ftm and the response file is .log. You can use these files in conjunction with the batch results to see which records were loaded successfully (with the internal id for those records), and which were not (with the error message for those records).

BLK_SAVE_RESPONSE_LOG

When set to *Yes*, saves the response log file. The log file will automatically be saved if BLK_SAVE_DATA_FILES is set to *Yes*. The default value is *No*. The file is saved in the same application as the flow and has the file type .log.

Creating Synonyms With Salesforce.com

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represents the server metadata.

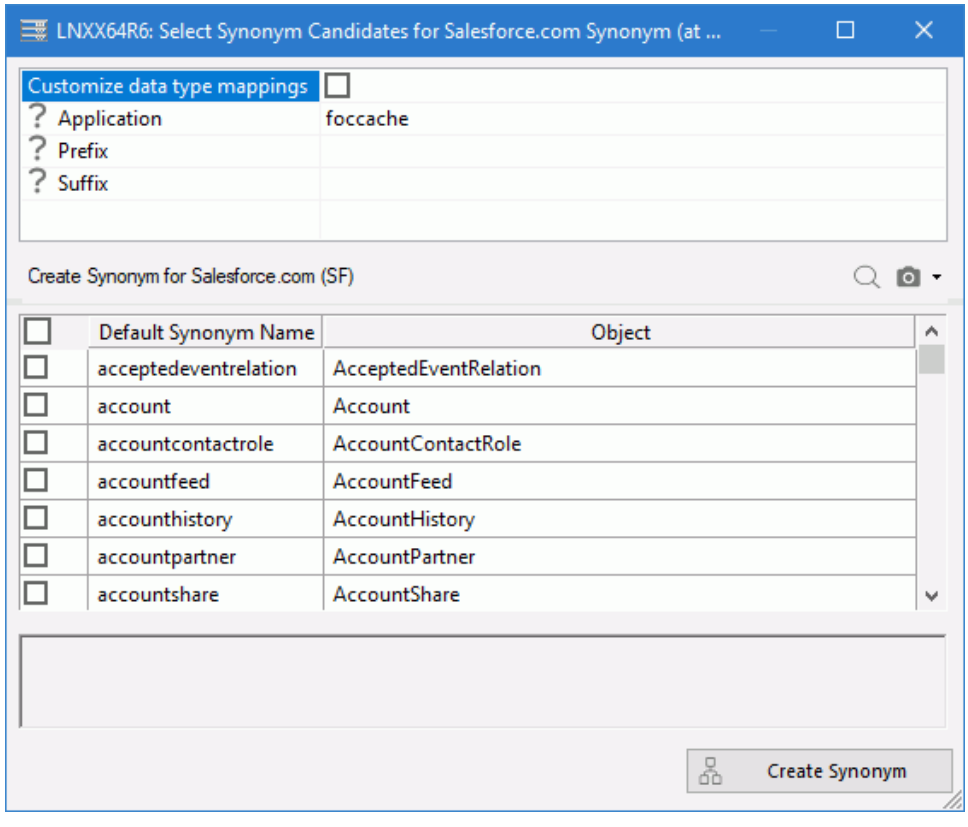
Procedure: How to Create a Synonym for a Salesforce.com Object

1. From the Data Management Console, right-click *Application Directories*, point to *New*, and then click *Synonym (Create or Update)*.

   The Select adapter to configure or Select connection to create synonym dialog box opens.

2. Select a connection for the configured adapter and click *OK*.

   The first of a series of synonym creation panes opens, showing a list of all of the available Salesforce.com objects, as shown in the following image.



3. Select the check box for the objects that you want to create synonyms for.

   If you want to change the name of the synonym from the default name, click the name and edit as needed.

4. Click *Create Synonym*.

   The Status pane indicates that the synonym was created successfully.

The synonym is created and added under the specified application directory.

Note: You can also create a synonym from the Adapters page by right-clicking a configured adapter connection and clicking *Create Synonym*.

Procedure: How to Create a Cluster Join for Salesforce.com Objects

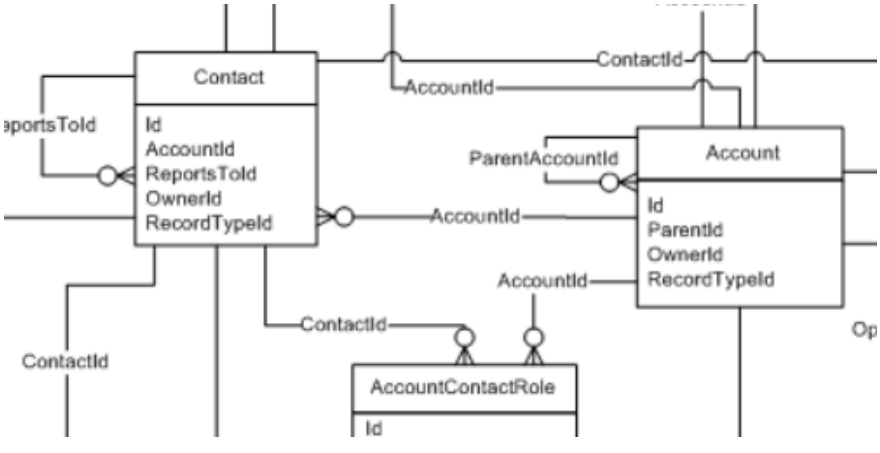
A synonym with a cluster join facilitates reporting against multiple salesforce.com objects.

Salesforce.com relationships between objects are described in the data models on the web site. Search by data model for a list of the entity relationship diagrams (ERDs) available. For example, the Sales Objects model includes accounts, contacts, opportunities and other related objects. For an example of the Sales Objects model, go to the following Web site:

http://www.salesforce.com/us/developer/docs/api/Content/sforce_api_erd_majors.htm

When you create a cluster join for salesforce.com objects, you must make it a subset of the data model. This allows the join between the objects to be performed by salesforce.com, minimizing the number of queries sent to their servers.

For example, say you wanted to create a synonym that would allow you to report from the contact object to get the first and last name of a contact, and the account object to get the company name. From the model shown in the following image, we can see that there is a relationship *from* Account to Contact.



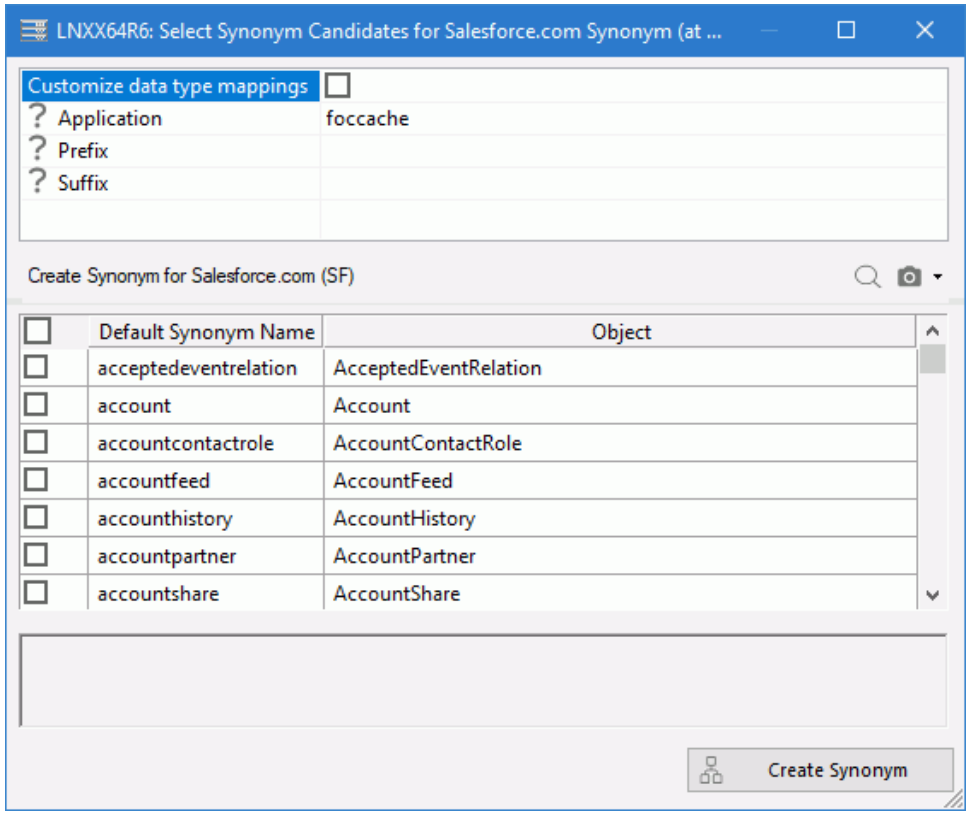
Here are the steps to create a synonym with a cluster join with that relationship.

1. Right-click an application directory, point to *New*, and then click *Synonym (Create or Update)*.

   The Select … connection to create synonym dialog box opens.

2. Select your connection to Salesforce.com and click *OK*.

The Select Synonym candidates... dialog box opens, as shown in the following image.



3. Select the check box in front of Account, Contact, and any other objects you want.
4. Click *Create Synonym*.

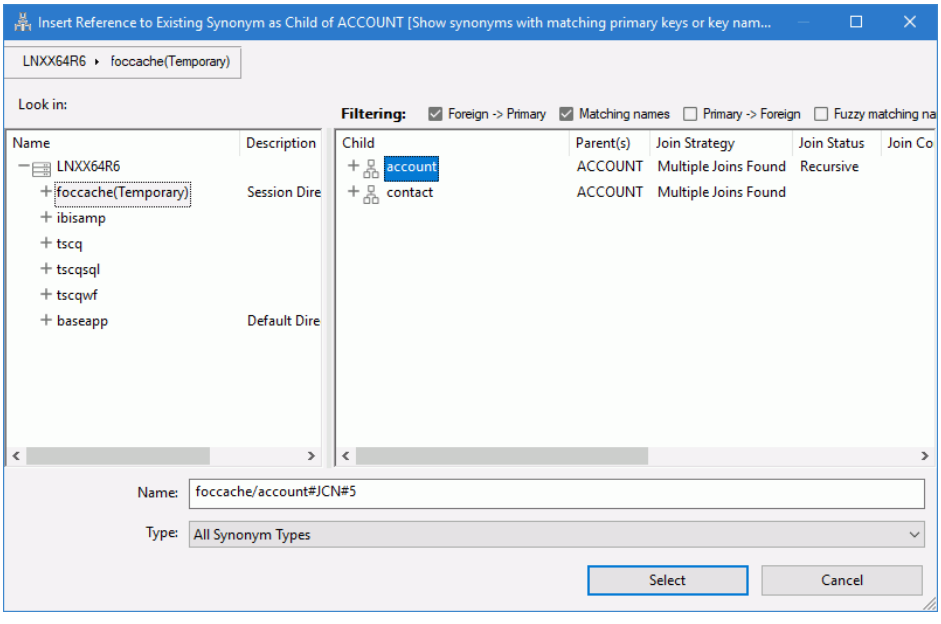
   The Status dialog box opens.
5. Ensure that for each synonym you selected, the message *Created Successfully* appears.

   Click *Close*.
6. Right-click an application directory, point to *New*, and then click *Synonym via Synonym Editor*.

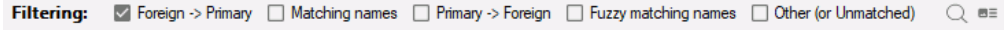
   A blank synonym opens.
7. Click the *Modeling View* tab. Right-click the workspace, point to *Insert*, and then click *Reference to Existing Synonym as Root*.
8. Click *account* and then click *Select*. The Account object is added to the workflow.

9. Right-click *account*, point to *Insert*, and then click *Reference to Existing Synonym as Child*.

   The Insert Reference to Existing Synonym as Child of ACCOUNT dialog box opens, as shown in the following image.

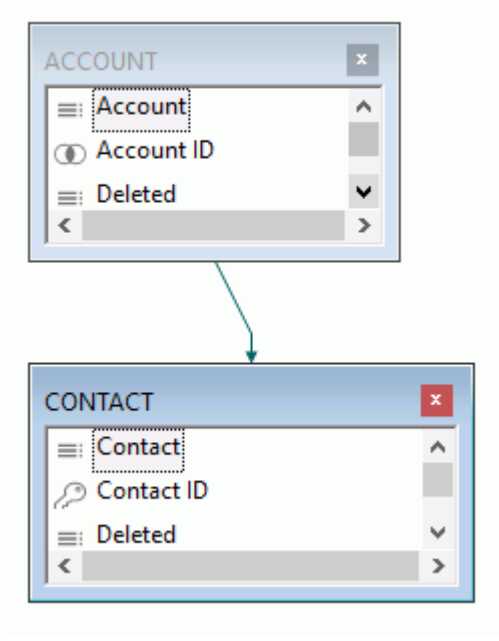


10. Clear all of the FIltering checkboxes except for Foreign -> Primary, as shown in the following image.



11. Click *contact* and then click *Select*.

The two objects can now be seen in the modeling view.



12. Click the *Save* button on the toolbar.

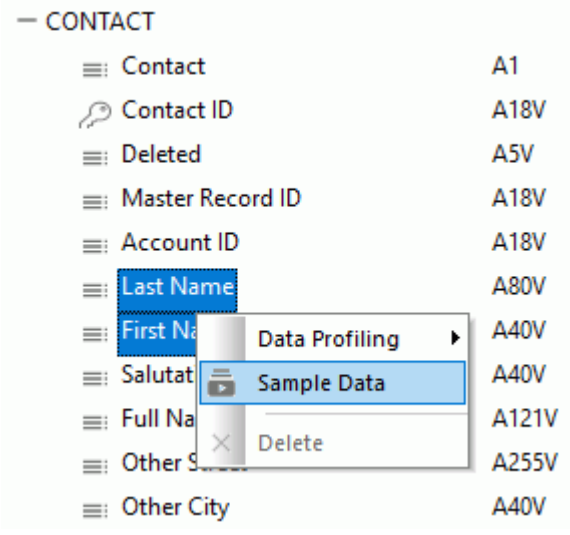
    The Save Synonym As dialog box opens.

    Enter *account_contact* for the name of the synonym and click *Save*.

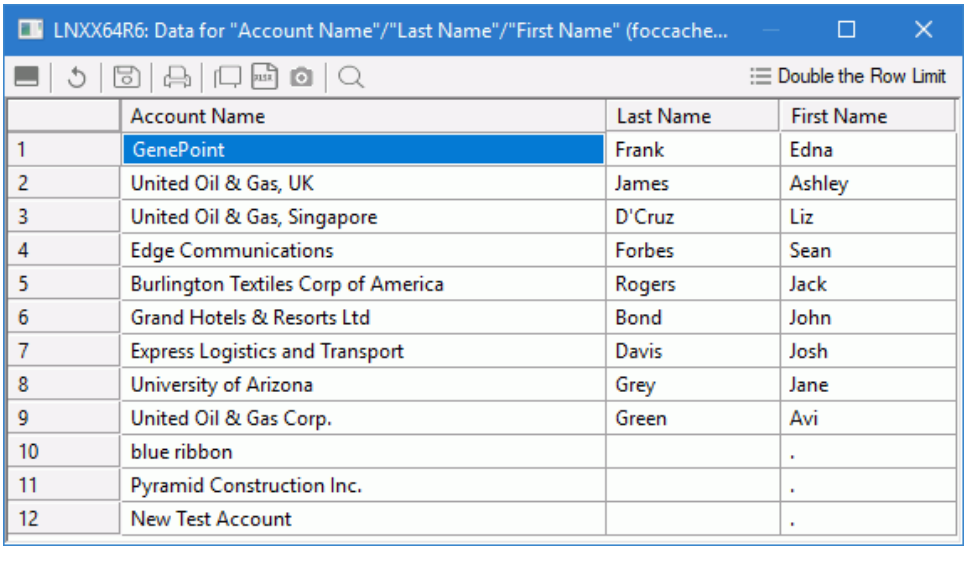
13. Click the *Field View* tab at the bottom of the workspace to switch the view.

14. Under the ACCOUNT segment, click *Account Name* and then scroll down to the CONTACT segment and Ctrl+click on *Last Name* and *First Name*.

15. After making the selections, right-click and select *Sample Data*, as shown in the following image.



A report opens showing data for Account Name, and the contact Last Name and First name, as shown in the following image.



	Account Name	Last Name	First Name
1	GenePoint	Frank	Edna
2	United Oil & Gas, UK	James	Ashley
3	United Oil & Gas, Singapore	D'Cruz	Liz
4	Edge Communications	Forbes	Sean
5	Burlington Textiles Corp of America	Rogers	Jack
6	Grand Hotels & Resorts Ltd	Bond	John
7	Express Logistics and Transport	Davis	Josh
8	University of Arizona	Grey	Jane
9	United Oil & Gas Corp.	Green	Avi
10	blue ribbon		.
11	Pyramid Construction Inc.		.
12	New Test Account		.

Syntax: How to Control the Retrieval of Deleted Records

You can control whether or not deleted and archived records are retrieved from salesforce.com.

```
ENGINE SFDC SET SHOWDELETED{YES|NO}
```

where:

`SFDC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`YES`

Deleted and archived records are included in the records displayed.

`NO`

Deleted and archived records are not retrieved. This is the default.

Syntax: How to Control the Block Size

You can control the number of records retrieved from salesforce.com in one block.

```
ENGINE SFDC SET BATCHSIZE nnnn
```

where:

`SFDC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`nnnn`

The number of records to retrieve in a block. The default is 200. The maximum is 9999.

Syntax: How to Use Salesforce.com Date Literals and Constants

Salesforce Object Query Language (SOQL) provides useful date constructs, for which there are no SQL equivalents. They can be used in a filter in a SELECT statement, or a TABLE request using the DB_EXPR function.

The following is an example of date constructs being used in a TABLE request using the function DB_EXPR:

```
WHERE datefield GT DB_EXPR(LAST_YEAR)
WHERE datefield GT DB_EXPR(LAST_N_YEARS:3)
WHERE (DB_EXPR(CALENDAR_MONTH("datefield")) EQ 1)
```

For more information on these date constructs, see *Date Formats and Date Literals* in the Salesforce.com documentation.

Chapter 90

Using the Adapter for SAP Business Intelligence Warehouse (BW)

The Adapter for SAP BW allows WebFOCUS for SAP Business Intelligence Warehouse (BW) and other applications to access SAP BW data sources through the SAP BW multi-dimensional (OLAP) model. The adapter converts data or application requests into native SAP BW statements and returns optimized answers sets to the requesting program.

In this chapter:

Preparing the SAP BW Environment

SAP BW remote communications require:

- SAP BW BASIS 6.4 or higher and SAP BW 3.1C or higher.
- Installation of the SAP RFC/SDK 4.6 or higher on the Server system.

  The Server requires the SAP RFC SDK to communicate with the SAP Application Server. The location of the RFC SDK depends on the platform specific search path.

  Platform dependent search path for shared libraries (RFC SDK):

  - **Windows:**
    - Directory where the executable is located.
    - Current directory.

- Windows system directory and Windows directory.
- Environment variable PATH.

- **AIX:**

  Environment variable LIBPATH.

- **On z/OS:**

  Environment variable LIBPATH.

- **All other UNIX platforms:**

  Environment variable LD_LIBRARY_PATH.

  If the server does not start, and indicates that some NLS libraries cannot be found, try:

  ```
  EXPORT NLSUI_7BIT_FALLBACK=YES
  ```

- **SAP BW 3.5 users.** SAP Note 11682 has been implemented. Required environment settings include:

  On all platforms (RFCSDK points to the SAP RFC SDK Shared libraries):

  ```
  SAP_HOME=$RFCSDK/lib
  ```

  On Solaris (for other platforms, please read OSS note 11682 for defining path):

  ```
  LD_LIBRARY_PATH=$SAP_HOME
  ```

  If the server does not start, and indicates that some NLS libraries cannot be found, try:

  ```
  EXPORT NLSUI_7BIT_FALLBACK=YES
  ```

- **Support for IBM iSeries platform.** This release introduces support for RFCSDK 640E as of OS Release V5R2 on SAP R3 and BW. When configuring the adapter, the RFCSDK 640E must be specified/added to the current library list.

  Related adapter configuration options for SAP BW on IBM platforms can be found in *How to Configure the Adapter on IBM Systems Using EBCIDC Character Sets* on page 2142 and *SAP BW Adapter Supporting Mixed Code Page Environments* on page 2142.

- A TCP/IP connection to the SAP BW source system.

The following SAP Authorization profiles are required at a minimum. This list may change depending on your SAP BW Release and/or site specific Authorization Profiles.

Reference: List of Authorization Objects, Fields, and Required Values

S_RFC: Authorization check for RFC access

Field	Value
ACTVT (Activity)	16 (execute)
RFC_NAME (Name of RFC to be protected)	RSAB, RSOB, SYST,BAPI_IOBJ_GETDETAIL,RSBAPI_IOBJ
RFS_TYPE (Type of RFC object to be protected)	FUGR,FUNC (function group)

S_RS_ADMWB: Administrator Workbench - Objects

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSADMWBOB (Administrator Workbench object)	INFOOBJECT

S_RS_COMP: Business Explorer - Components

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSINFOAREA (InfoArea)	* (All or specific InfoAreas)
RSINFOCUBE (InfoCube)	* (All or specific InfoCubes)
RSZCOMPID (ID of a reporting component)	* (All)
RSZCOMPTP (Component type)	CKF, REP, RKF, STR, VAR

S_RS_COMP1: Business Explorer - Components: Enhancements

Field	Value
ACTVT (Activity)	03 (Display), 16 (Execute)
RSZCOMPID (Name (ID) of a reporting component)	* (All)
RSZCOMPTP (Type of a reporting component)	CKF, REP, RKF, STR, VAR
RSZOWNER (Owner (Person Responsible) for a Reporting Component)	* (All)

S_RS_ICUBE: Administrator Workbench - InfoCube

Field	Value
ACTVT (Activity)	03 (Display)
RSICUBEOBJ (InfoCube Subobject)	DATA
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSINFOCUBE (InfoCube)	* (All) or specific InfoCubes

S_RS_ISET: Administrator Workbench -InfoSet

Field	Value
ACTVT (Activity)	03 (Display))
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSINFOSET (InfoSet)	* (All) or specific InfoSets
RSISETOBJ (InfoSet-Subobject)	DATA

S_RS_ODSO: Administrator Workbench - ODS Object

Field	Value
ACTVT (Activity)	03 (Display))
RSINFOAREA (InfoArea)	* (All) or specific InfoAreas
RSODSOBJ (ODS Object)	* (All) or specific ODS Objects
RSODSPART (Subobject for ODS Object)	DATA

Accessing Multiple Systems

The query adapter can operate across multiple BW systems. For each system, the query adapter requires one BW logon consisting of client, user, and password. This logon must be:

- RFC-enabled.
- Authorized for all clients that you may access.

Authorizations in SAP-BW are specific to the local security policy. In addition to the authorizations documented in this manual, additional authorizations will be required in most instances. Please contact your SAP-BW basis administrator.

Configuring the Adapter for SAP BW

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

In addition, for SAP BW running on z/OS, you must install the SAP Code Page on the BW application server.

Declaring Connection Attributes

In order to connect to the SAP BW Application Server, the adapter requires connection and authentication information, which is supplied using the SET CONNECTION_ATTRIBUTES command.

You can enter connection and authentication information on the Configuration pane of either the Web Console or the Data Management Console. The global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile will be updated with the adapter connection attributes.

You can declare connections to more than one SAP BW data source by including multiple connections. The actual connection to SAP BW takes place when the first query that references the connection is issued. If more than one connection is declared, the adapter uses the attributes specified in the *last* connection.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for SAP BW

The *SAP BW* adapter is under the *OLAP* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

The major work of the query adapter is to translate the user request into code that can be understood by SAP BW.

System

Name of the connection; maximum 12 characters.

Connection Parameters

The Load Balancing check box determines which of the following options is exposed.

GROUP

Name of the application group. An application group defines a list of application servers on which an RFC application can be running. R/3 transaction SMLG can be used to view or modify application groups.

Note that the entries are case-sensitive and blank spaces are significant.

Appears only when Load Balancing is checked.

MSGHOST

Host name of the SAP system (message server).

Appears only when Load Balancing is checked.

R3NAME

System ID of the SAP system.

Appears only when Load Balancing is checked.

HOST

Host name of the SAP application server.

GWHOST

Host name of the machine where the SAP gateway process is running. In the case where there is only one SAP application server, gwhost and host is the same.

SYSNR

SAP system number. This is a 2-digit numeric value. Obtain this value from the SAP Administrator.

CONNECTION LANGUAGE

For SAP BW Unicode configurations, you can select a language that is different from the language you use when logging on to the server (as determined by your NLS configuration).

From the CONNECTION LANGUAGE drop-down list on the Add SAP BW non Unicode to Configuration pane, select the language you wish to use to connect to SAP BW.

Note:

- When the CONNECTION LANGUAGE option is used to specify a language other than the one used for logging in to the server, the Create Synonym Multilanguage option is not available.
- For this option to work, the codepage of the iWay server must match the codepage of the SAP BW server. For example, if the iWay server is configured FOR codepage 942 (SJIS), the SAP server must be configured for codepage 8000.
- There is no way to check if the requested connection language can be used by either the iWay server code page (either in terms of display or number of bytes) or the SAP BW server.

SAP Security Mode

When SNC is checked, you must specify values for the following additional parameters:

SNC_LIB

SNC_LIB contains the path to the external security product library. The external security product's library, external library, SNC_LIB, or gssapi library contains the functions provided by the external security product certified by SAP.

Set the environment variable SNC_LIB to contain the path to the security product library:

```
<drive>:\path\to\your\snclib.dll
```

SNC_ PARTNERNAME

Is the external name of the SAP system. This is an extended version of the external name called the SNC name. You create the SNC name by providing a prefix with the external user name that designates the name type entered as follows:

```
<SNC-name_of_SAP_AppServer>
```

For example:

```
p/secude:CN=miller,
OU=ADMIN, O=SAP, C=DE

p/krb5:miller@WDF.SAP-AG.DE
```

SNC_QOP

Indicates the level of protection.

`SNC_MYNAME`

Indicates the SNC name of the initiator, as in *own_snc_name*.

Note: SNC with load-balancing. Load-balancing, or group-logon, dynamically retrieves the target SNC-Name from the message server. When using SNC with load balancing, you must specify the following additional parameters:

```
SNC_PARTNERNAME=p:unused
SNC_LIB=<drive>:\path\to\your\snclib.dll
```

Click *Configure*.

The next step is adding a connection. Click *Next* to continue.

General User Login Parameters

Security

There are three methods by which a user can be authenticated when connecting to an SAP BW instance:

- **Explicit.** The user ID and password are specified for each connection and passed to SAP BW for authentication and request execution. (When you select Explicit, the USER and PASSWORD options are displayed.)
- **Password Passthru.** The user ID and password received from the client application are passed to SAP BW for authentication and to execute the user request.
- **Trusted.** Is set automatically when SNC is selected.

CLIENT

SAP BW Client for the user logon; maximum three characters.

USER

SAP BW user ID for the user logon.

PASSWORD

SAP BW password for the user logon; maximum 40 characters.

For more information, see *Preparing the SAP BW Environment* on page 2133.

Note: If the SAP BW system is running in a z/OS (USS) environment, see *How to Configure the Adapter on IBM Systems Using EBCIDC Character Sets* on page 2142.

Procedure: How to Configure the Adapter on IBM Systems Using EBCIDC Character Sets

To configure the adapter on IBM systems using the EBCIDC character set, the following code pages must be installed on the BW application server:

- In a z/OS environment, SAP code page 0126.
- For iSeries, SAP code page 0123.

1. Create two conversion tables as described in *SAP BW Adapter Supporting Mixed Code Page Environments* on page 2142, and transfer the tables to the $EDACONF/etc directory.

   For example, if the Server environment uses the code page 1100, then two conversion tables, 11000126.CDP and 01261100.CDP, should be created and transferred to the $EDACONF/etc directory.

2. The following environment variables are required at run time:

   ```
   export SAP_CODEPAGE=0126
   export PATH_TO_CODEPAGE=$EDACONF/etc/
   ```

   Note: Make sure to include the trailing "/" for the value of the PATH_TO_CODEPAGE parameter.

SAP BW Adapter Supporting Mixed Code Page Environments

If you are using an SAP server that supports unicode, the adapter must also be configured for unicode. With this configuration, no character conversion tables are required.

If, however, you are using an older SAP server that does not support unicode the following information applies. SAP BW, as well as adapters, can be installed on various platforms. If SAP BW is installed on a platform that uses a code page or character set different from the adapter platform, two character conversion tables must be generated to translate the different character sets, one for each direction of data flow. For example, suppose that SAP BW is installed on an IBM i EBCDIC machine and the Adapter for SAP BW is installed on an Intel Windows ASCII machine. If a request is going from the server to SAP BW, an ASCII to EBCDIC translation is required. If SAP BW is sending data back to the Adapter for SAP BW, an EBCDIC to ASCII translation is required. This section describes how to create conversion tables and where they must reside.

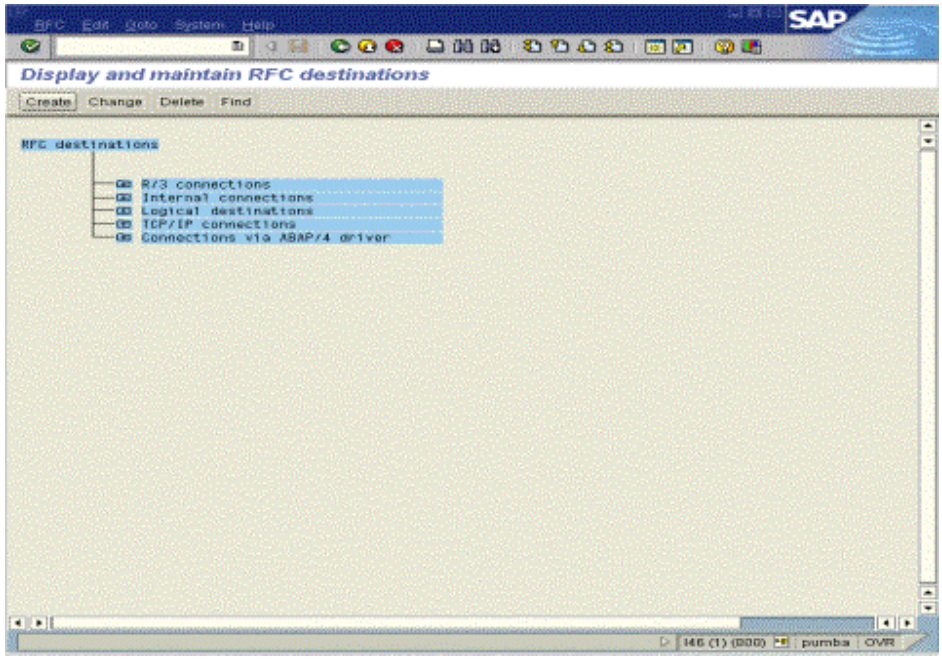
Character Conversion Tables

SAP BW provides a transaction, SM59, to generate conversion tables for RFC-based applications such as the Adapter for SAP BW. Once these tables are created, they must be copied to the adapter.

Procedure: How to Generate Conversion Tables

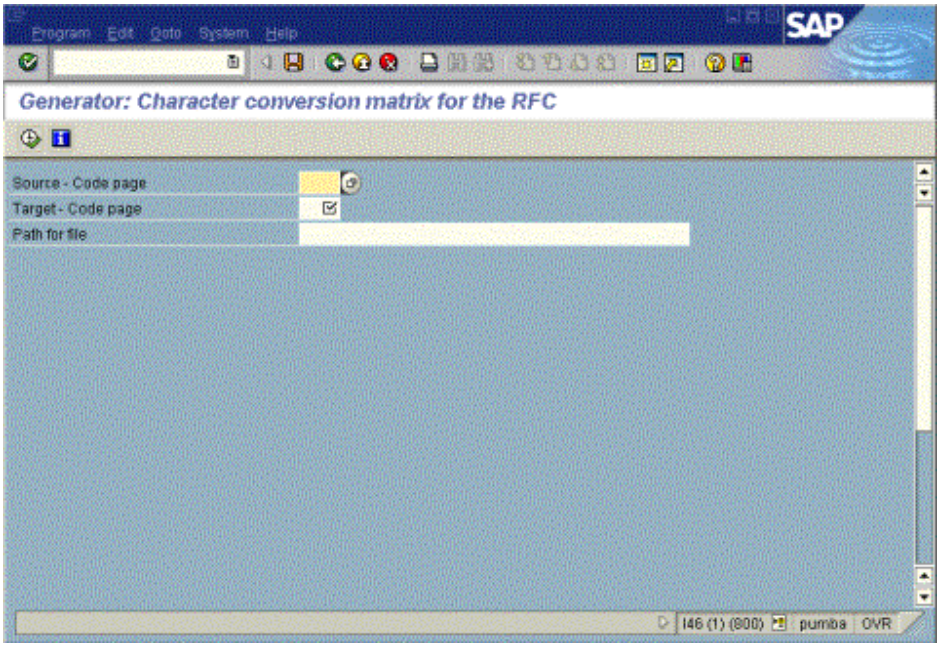
1. Type transaction SM59 at the SAP GUI command line.

   The following window opens.



2. Select the *Generate Conv.* Tab option from the RFC pull-down menu.

   The following window opens.



3. Type a value for the Source code page, Target code page, and a valid path on the application server where the conversion table file is created. Optionally, you can supply a valid path and a unique file name.

4. Click *Execute* .

   If you did not specify a unique file name, the default file name of the newly created conversion table will contain the source and target code page values. For example, if you specified a source code page of 0102 (IBM i for some CUAs) and a target code page of 1101 (7 bit USA ASCII pur), the default file name is 01021101.cdp. This is your IBM i (EBCDIC) to ASCII conversion table file.

   It is now necessary to create another conversion table file for translating code pages in the opposite direction.

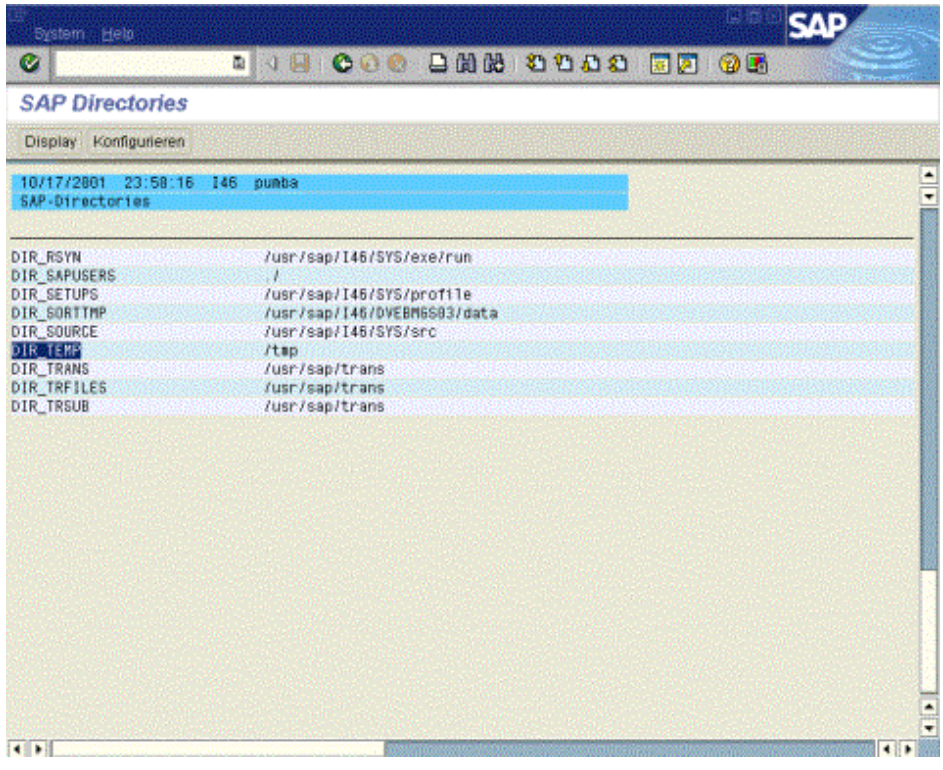
5. Switch the source and target code pages and generate the conversion table.

   Continuing with previous example, you should have a source code page of 1101 and a target code page of 0102, resulting in a default file name of 11010102.cdp. This is your ASCII to IBM i (EBCDIC) conversion table file.

Note: If a unique file name was used for the first conversion table, be sure you specify a unique file name for the second conversion table.

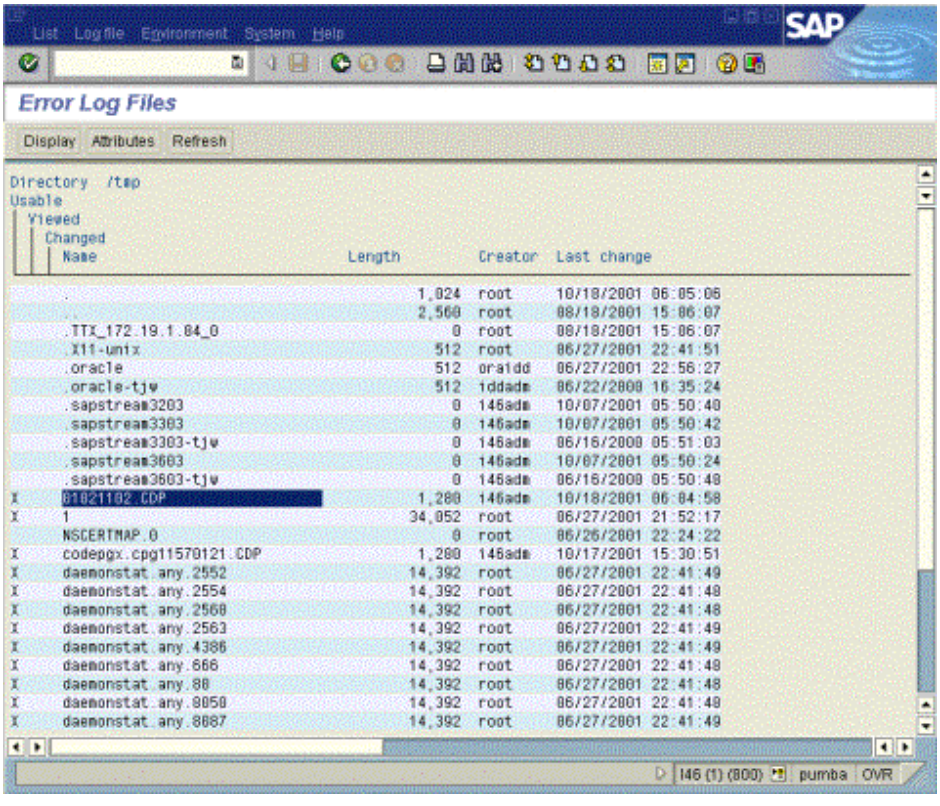
Procedure: How to Download the Conversion Tables to the Server

Once these conversion table files are created, you can download them from the SAP application server directory to the server configuration ETC directory. Use SAP BW transaction AL11 to browse and download the two conversion table files to a local file on the desktop or Windows machine where the server is installed. Note that SAP GUI is required on the machine to which you are downloading the conversion table files. The following window represents transaction AL11.



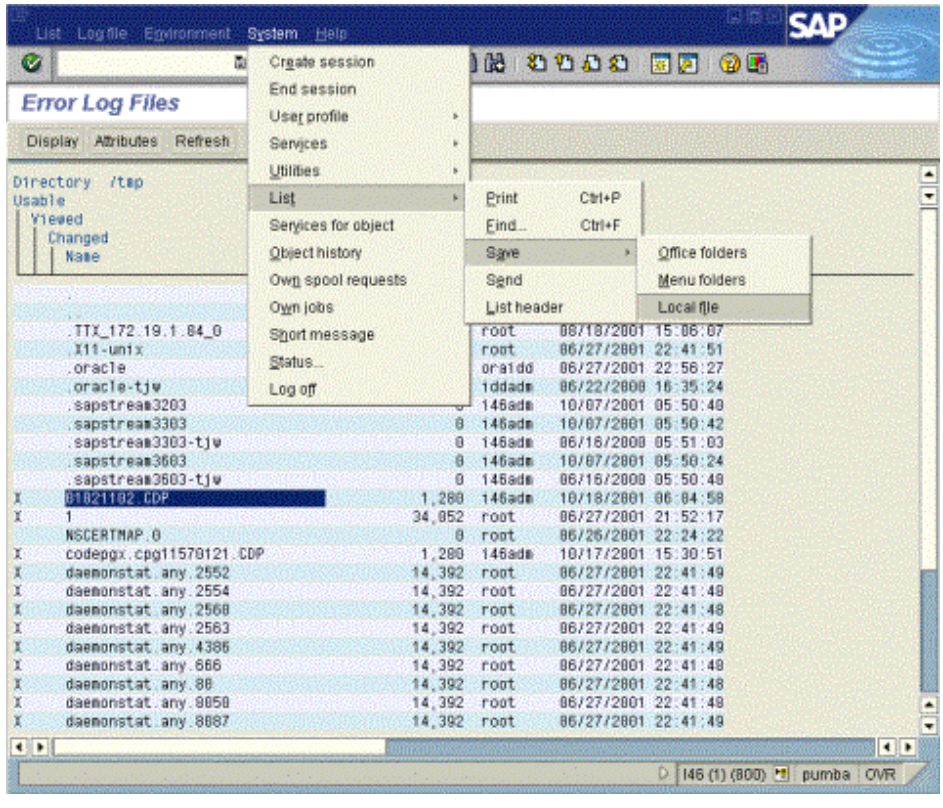
1. Double-click the directory where the conversion table files were created.

The following window opens.



2. Double-click the conversion table file to view its contents.

3. Select the List option from the System pull-down menu and select *Save to Local File*, as shown in the following image.



A dialog box displays prompting on a format for the transfer file.

4. Select unconverted format and click *Continue*.
5. Type a valid file name and click *Transfer*. This transfers the file to your local desktop.
6. Use any available editor on your desktop and remove the first three lines from the conversion table file. The following is an example of the text that should be removed:

```
Directory /tmp
Name: 01021102.CDP
------------------------------------------------
```

After the edits have been made, configure the server with the Adapter for SAP BW. Once the adapter is configured, the conversion table files must be moved to the server EDAHOME ETC directory. Make sure both conversion table files are moved.

Creating BEx Queries

The adapter can report against several types of SAP BW objects, some of which represent views or subsets of the actual data. Using the Business Explorer (BEx) provided by SAP as its reporting and analysis tool, you can create queries (methods for extracting data) and reference them in WebFOCUS report requests.

A query is a subcube created by selecting characteristics and key figures. Using queries, the data can be quickly targeted and evaluated. The more precisely the query is defined, the smaller the subcube and the quicker the query can be navigated and refreshed. Selecting certain characteristics means that they can be more closely analyzed while others remain unspecified. The resulting key figures are aggregated across all characteristic values for the unspecified characteristics.

A default navigational state is also established in the query definition when you arrange the characteristics and key figures in the rows and columns of the query.

Only the data that is currently requested is transferred to the query. The OLAP processor builds the query from the data and provides methods for navigating through the data in several dimensions. Since a query preselects information, the same data can yield dramatically different results depending on the query used to view its contents.

SAP BW Terminology

The Business Explorer (BEx) is the SAP BW component that provides reporting and analysis tools. BEx queries filter the data to create reporting objects or views called InfoProviders. Some InfoProviders contain actual data. Others provide views of data stored in elsewhere.

The basic elements used to define InfoProviders are called *InfoObjects*. An InfoObject is a business evaluation object, such as *customer* or *sales revenue*, relevant to your analysis of business performance. InfoObjects can be *characteristics* (dimensions), *key figures* (measures or facts), *units*, *time characteristics*, or *technical characteristics* (for example, request number).

The following InfoProviders can be used for WebFOCUS reporting:

- **InfoCube.** An InfoCube is a self-contained set of business data. The physical data store is called the BasicCube. It consists of InfoObjects and is structured in a star schema, which means that one large fact table contains the key figures (numeric data values). This fact table is surrounded by smaller dimension tables that store characteristics (categories of data). Other types of InfoCubes display logical views of a data set. Some examples follow.
  - **RemoteCube.** A RemoteCube is an InfoCube whose transaction data is not managed by BW, but externally

- **SAP RemoteCube.** An SAP RemoteCube is a RemoteCube for which you can define queries by accessing transaction data in other SAP systems.
- **Virtual InfoCube with Services.** A Virtual InfoCube with Services is an InfoCube whose data is not stored in BW and for which a user-defined function module is used to retrieve the data.

- **ODS Object.** An ODS object is a set of cleaned up transaction (detail level) data. It contains key fields (for example, item number) and data fields (such as order status or customer).
- **InfoSet.** An InfoSet is a join between ODS objects and/or InfoObjects. It is a definition of the conditions that determine how the objects should be joined.
- **MultiProvider.** A MultiProvider is a combination of data from separate InfoProviders. It does not contain data; data comes from the separate InfoProviders.

An **InfoObject catalog** is an application-specific organization of InfoObjects, either characteristics or key figures.

To report against a cube, you must create a BW synonym that describes it to the adapter. To create a synonym for a query cube, you first create the BEx query. In the query properties, set the *Extended Query Properties* to *Allow external access to this Query*.

BEx Query Terminology

The basic building blocks of a query cube are:

- **Characteristics.** Characteristics, also called *dimensions*, are classifications such as Region or Time, that you use to analyze and compare business performance.
- **Attributes.** Attributes, also called *properties*, are additional information about a characteristic. For example, the *material* characteristic might have attributes such as *color* or *weight*.
- **Key Figures.** Key figures, also called *measures* or *facts*, are quantifiable values used in evaluating performance. An example of a key figure is *sales revenue*.
  - **Calculated Key Figures.** These key figures are not stored in the data. They are defined in the BEx query and evaluated at run time.
  - **Restricted Key Figures.** These key figures are filtered by selecting one or more characteristics. They can be filtered versions of basic key figures, calculated key figures, or other restricted key figures.

- **Hierarchies.** Hierarchies are groupings of characteristics according to individual evaluation criteria. For example, a *customer* characteristic can be grouped by geographical location, industry, or some other criteria. A characteristic can have multiple hierarchies.
- **Variables.** Variables are parameters of a query. They are set in the query definition and may get their values in different ways. For example, some variables are given default values and others require the user to enter a value. For more information about variables, see *Variable Types* on page 2223 and *Reporting With Variables* on page 2197.
- **Filters.** Filters are used to restrict the output of a request. Although not included in a WebFOCUS synonym, they affect the outcome of a WebFOCUS request.

Defining New Business Explorer Queries

You must define an SAP Business Explorer query before reporting from an SAP BW InfoProvider. This query serves as a template for data extraction from the cube.

Note: The information that follows is based on SAP BW Business Explorer documentation. SAP BW BEx documentation is available from *http://help.sap.com*.

Procedure: How to Select an InfoProvider to Query

1. The Business Information Warehouse must contain at least one InfoProvider before you can define a new query. Start the Business Explorer Analyzer.
2. From the BEx toolbar, choose *Open*.

   You will see the selection screen for all existing workbooks.
3. Choose *Queries*.

   The selection screen displays all available queries.
4. Choose *New*.

   You will see the selection screen for all InfoProviders for which you can define a new query or queries.
5. Select the InfoProvider that has the data on which the query should be based. To display technical names for InfoProviders, set the Technical name on/off icon to *On*.

   The available objects in the InfoProvider you have selected are displayed as a directory tree in the left part of the screen.

Next, you will select the objects for the query and drag them to the appropriate boxes to build the query.

Procedure: How to Create a Query

The right area of the BEx screen contains selection boxes for the filter selection, the rows, the columns, and the free characteristics of the query.

Perform the following steps to create a query:

1. Click the plus or minus sign to the left of the dimension or Key Figures you want to query.

   The object list will expand and display a list of all the available Key Figures or characteristics.

2. Drag and drop characteristics and Key Figures from the InfoProvider into the selection box of the query definition.

   These may be filters, rows, columns, and free characteristics.

Procedure: How to Filter a Query

You can filter queries in order to place restrictions on them. Filter selection restricts the entire query. To select fields you want to use to filter the query, complete the following:

1. From the object list of the InfoProvider, select the characteristics or the key figure upon which the query should be based.

   Note: Since they are used in definitions, fields selected as filters are not displayed in the Adapter for SAP BW metadata. They are used to screen the data and thus contain no information to be reported on. If you wish to screen data and report from it, see *Restricting Query Characteristics* on page 2151 or *Restricting and Calculating Key Figures* on page 2152.

2. Drag the object to the Filter box.
3. Right-click the object in the filter box. A dialog box opens showing the possible filter definitions for the object.
4. Select either a single member, a range of members, or a variable for the filter.

Restricting Query Characteristics

When defining a query, you may restrict characteristics to a single characteristic value, a value interval, a hierarchy node, or a characteristic value variable.

Procedure: How to Restrict Characteristics

1. Choose the characteristic from the InfoProvider for which you want to select a value range.

2. Drag the characteristic into the appropriate selection box of the query definition (rows or free characteristics).

3. Select the characteristic you wish to restrict (or filter). Using the right mouse button, select *Restrict* from the Context menu.
4. Choose whether you want to restrict the characteristic to a single value, a value interval, or a hierarchy node.

   Tip: You can enter the characteristic values or hierarchy nodes you want to use, or you can display a list of all possible values by clicking the magnifying glass to the right of the input field.
5. Confirm your entries by clicking *OK*.

Restricting and Calculating Key Figures

You can restrict Key Figures to characteristic values, characteristic value intervals, or hierarchy nodes. For example, a restricted key figure would be Sales revenue in 1st quarter.

You can also restrict the Key Figures of the InfoProvider for the query definition, or, using a formula, you can calculate new Key Figures from the (basic) Key Figures:

- **Restricted Key Figures.** (Basic) Key Figures for the InfoProvider that are restricted (filtered) by selecting one or more characteristics.
- **Calculated Key Figures.** Formulas that consist of (basic) Key Figures for the InfoProvider and/or calculated Key Figures that have already been created.

Procedure: How to Restrict Key Figures

1. Drag a (basic) key figure into the key figure selection box. Alternatively, select the header of the selection box for rows or columns and, using the right mouse button, select *New Structure* from the Context menu.
2. Select the Structure directory, and, using the right mouse button, choose *New Selection* from the Context menu. The New Selection screen opens.
3. Enter a description of the restricted key figure in the text fields located in the upper part of the screen.
4. Underneath the text fields, on the left, is the directory of all the objects available in the InfoProvider. Use the empty field on the right-hand side of the screen for the definition of the new selection.
5. Using drag and drop, choose a key figure from the InfoProvider, and restrict it using a selection of one or more characteristic values.
6. Select *OK*. The newly restricted key figure is defined in the structure.

Procedure: How to Calculate Key Figures

1. Create a new structure in the rows or columns of the query definition by highlighting the row or column directory using the right mouse button and selecting *New Structure* in the Context menu.
2. Drag a (basic) key figure of the InfoProvider into the directory of the new structure.
3. Select the Structure directory, and choose *New Formula*. The Formula Definition screen opens.
4. Enter a description of the formula in the text fields located in the upper part of the screen.

Note: The entry field for the formula is underneath the text fields. In the bottom left of the screen are all of the operands available for the formula definition. These are the Key Figures that you have already defined in the structure, and all of the formula variables in the Variables directory that have been created in the variable maintenance.

The functions available as operators are on the right-hand side of the screen. These are symbols for the basic arithmetic operations and directories with calculation functions such as percentage or trigonometric functions. To the right of the operators is a number block.

Procedure: How to Define a Formula

1. Choose the operands you want to use, and insert them in the entry field for the formula by double-clicking or by using drag and drop.
2. Choose the calculation functions you want to use by either clicking the symbols for the basic arithmetic operations, double-clicking to select the individual values, or dragging the entire key figure into the formula box.
3. Select the number values for the formula by clicking the number block.
4. Define your formula using the available operands and operators.

   If you want to use a variable that is not contained in the operands, you must create the variable first.
5. Check the formula definition for correctness by pressing the scale icon.
6. Enter the name of the formula column in the description box.
7. Select *OK*. The newly calculated key figure is defined in the structure.

Viewing Query Properties and Releasing for OLAP

- To view the properties of a query, click the Query Properties icon on the toolbar. The Query Properties dialog box opens.

- To release a query for OLAP, click the Query Properties icon on the toolbar and check the following in the Query Properties dialog box:
  - *Allow External Access* (for SAP BW releases 3.x and higher).
  - *Release for OLE DB for OLAP* (for SAP BW releases prior to 3.x).

This enables the query to be displayed as a QUERY_CUBE for reporting purposes. The query elements (hierarchy levels, measures, variable, and properties) will be mapped to corresponding OLAP elements to create a synonym.

SAP BW Reporting With WebFOCUS

Overview of SAP BW Reporting Concepts

In a multi-dimensional data source (cube), dimensions (called *characteristics* in SAP BW) are categories of data, such as Region or Time, that you use to analyze and compare business performance. Dimensions consist of data elements that are called *members*. For example, a Region dimension could have members *England* and *France*.

Dimension members are usually organized into hierarchies. Hierarchies can be viewed as tree-like graphs where members are the nodes.

For example, the Region dimension may have the element *World* at its top level (the root node). The World element may have children nodes (members) representing continents. Continents, in turn, can have children nodes that represent countries, and countries can have children nodes representing states or cities. Nodes with no children are called *leaf nodes*.

Measures (also called *key figures* in SAP BW) are numerical values, such as Sales Volume or Net Income, that are used to quantify how your business is performing.

A multi-dimensional cube consists of data derived from facts, which are records about individual business transactions. For example, an individual fact record reflects a sales transaction of a certain number of items of a certain product at a certain price, which occurred in a certain store at a certain moment of time. The cube contains summarized fact values for all combinations of measures and members of different dimensions.

For example, the following combination (*tuple*) contains the total volume of sales of pumps in all stores in England in 2005:

```
{Sales Volume, Pumps, England, 2005}
```

The point in the cube that contains this summarized value is called a cell. A cell is addressed by a combination of members of different dimensions and a measure. In this example *Sales Volume* is a measure and *Pumps*, *England*, and *2005* are members of the Product, Region, and Time dimensions respectively.

Individual fact records are usually tied to the leaf nodes of each hierarchy in the cube. The fact values get included in cells addressed by these leaf nodes and added to all cells addressed by all combinations of ascendants of these leaf nodes along each hierarchy of the cube.

The operation used to summarize facts for some measures can be a simple sum or a more complex aggregation function such as an average.

It often happens that some combinations of hierarchy nodes do not have any fact records tied to them. The cells addressed by these combinations are empty cells.

As illustrated in the previous example, a tuple is a combination of members from different dimensions of a cube. The previous tuple contains members from all dimensions and, therefore, addresses a single cell. If a tuple contains only members from some dimensions, it addresses not just one cell but a whole slice of cells in the cube. For example, the following tuple does not include either Region or Time dimension members:

```
{Sales Volume, Pumps}
```

It addresses as many cells in the cube as there are members of the Region dimension times the number of members of the Time dimension.

The number of cells in the cube addressed by a single dimension member is a product of cardinalities of all other dimensions.

When all cells addressed by a member are empty, the member is called an empty member. When all cells addressed by a tuple are empty, the tuple is called an empty tuple.

By default, empty cells do not appear on report output. You can issue the following command if you want them on the report output:

```
ENGINE BWBAPI SET EMPTY ON
```

BW cubes can also include *variables*, which are parameters used in data selection. Variables cannot be displayed or used for sorting; they can only be used in selection criteria. A variable can be defined as mandatory or optional. If a variable is mandatory, the request must contain a WHERE test using that variable. The WHERE criteria must provide the type of restriction required based on the variable selection type, for example, a single value (using an equality test) or an interval (using a FROM/TO or GE/LE test).

When the adapter accesses a multi-dimensional cube, it uses two types of metadata elements about dimensions:

- **Hierarchy fields.** These fields contain data that apply to a specific hierarchy and describe each member's position within that hierarchy. For example, these fields identify the member's parent as well as its own name, caption, unique ID, and level number.

- **Dimension properties.** These fields contain data that potentially apply to all members of the dimension. For example, a Region dimension may have a property called GEOGRAPHICAL_HEIGHT that specifies the altitude for each member of the Region dimension.

Using the hierarchy fields, the adapter can recreate the hierarchy and locate portions of the hierarchy needed to satisfy a request.

Understanding Columnar and Hierarchical Reporting

Two types of hierarchy can be represented in a synonym: level and parent/child.

A synonym describes a level hierarchy by using a separate field for each level. To report on a level hierarchy, you use *columnar reporting* in which you specify the field name for each level you want to display.

A synonym describes a parent/child hierarchy using a set of fields that define the hierarchical structure and the relationships between the hierarchy members. The adapter has special *hierarchical reporting* syntax for reporting on parent/child hierarchies.

Note: When you create a synonym, you can choose whether to describe the parent/child hierarchies in this way or as level hierarchies, with a separate field for each level.

Hierarchical reporting enables you to sort and select members of parent/child hierarchies without knowing specific level numbers.

A hierarchical reporting request goes through several phases before output is displayed.

Hierarchical Sorting and Member Selection

The first phase selects hierarchy members to display. The hierarchical reporting phrase BY or ON HIERARCHY automatically sorts and formats a hierarchy with appropriate indentations that show the parent/child relationships. If you do not want to see the entire hierarchy, you can use the WHEN phrase to select hierarchy members for display. The expression in this WHEN phrase must reference only hierarchy fields, not dimension properties or measures.

Dimension properties and measures are linked to the leaf nodes of the dimension and, therefore, cannot be used in selecting hierarchy levels for display.

Screening Dimension Data

Once hierarchy members are selected, you can screen the retrieved dimension data by applying WHERE tests to the selected members.

WHERE criteria are applied to the leaf nodes and are processed after the phase of the request that selects hierarchy members. Therefore, dimension properties can be used in WHERE tests.

These tests can also reference hierarchy fields. However, since the selection criteria are always applied to the values at the leaf nodes, they cannot select data based on values that occur at higher levels. For example, in a dimension with Continents, Countries, and Cities, your request will not display any rows if you use WHERE to select at the Country level, but it may if you use it to select at the City level.

Screening Based on Aggregated Values

Measures, being summarized values, can be referenced in WHERE TOTAL tests and COMPUTE commands because those commands are processed after the hierarchy selection and aggregation phases of the request. When screening with WHERE TOTAL, the aggregation phase of the report processing is over, so totals on the report are not recalculated to account for the data that is screened out, the rows are just removed.

Reference: **Prerequisites for Hierarchical Reporting**

Hierarchical reporting uses special metadata attributes and reporting syntax. You must:

- ❑ Create a synonym that describes hierarchies as parent/child relationships.
- ❑ Use hierarchical reporting syntax in your request to automatically build and format hierarchies in the report output.

For more information on creating synonyms, see *Creating Synonyms* on page 2213. For more information on hierarchical reporting, see *Hierarchical Reporting* on page 2163.

Representing Hierarchies in a Synonym

Dimensions are organized into sets of hierarchies. For example, in a Time dimension, years, quarters, and months can form a hierarchical or parent/child relationship. This means that the measures for each month are aggregated into values for quarters, and the quarters are aggregated into values for years. Each point in the hierarchy is called a *node*. Nodes at the bottom of the hierarchy (with no children) are called *leaf nodes*.

According to the OLAP model, each dimension has one hierarchy called the *flat hierarchy*. This hierarchy contains all leaf node members of the dimension as the children of the root node named ALL (which is not assigned a field name in the synonym). In a synonym, the flat hierarchy is assigned the same name as the dimension.

In a synonym, hierarchies can be described in one of two ways:

- ❑ **Level hierarchy.** Each hierarchy level is described using a separate field name. To issue a report request, you must reference the field name for those levels you want to display on the report output.

When you create a synonym with level hierarchies, the synonym contains one field declaration for each level. This declaration also specifies which field is its parent. If the data changes to have an additional level, you must recreate the synonym in order to account for this additional field and parent reference.

- **Parent/child hierarchy.** The hierarchy is described with a set of fields that contain values for properties that describe each member position in the hierarchy. For example, there are fields to contain a member's unique ID, level number, its parent, and the parent member level number. To issue a report request, you only need to specify the field name of one of the hierarchy fields.

  With a parent/child hierarchy, one set of field names in the synonym describes the hierarchy. A change in the number of levels does not require a change to the synonym.

  Another advantage of parent/child hierarchies is that the adapter can recreate and format any portion of the hierarchy. The request does not have to specify level numbers.

  Note: The flat hierarchy is always defined as a level hierarchy with one level.

Dimension properties apply to all hierarchies in the dimension and are listed in the synonym following all of the dimension's hierarchies. Most of the examples in this topic use synonyms named ZOPT and ZLEVEL created using the cube 0SD_C01/ZTSCQ31CQ1.

Example: **Dimension Declaration**

Each dimension begins with a dimension record that defines the dimension and its hierarchies. The dimension itself is level zero. In this example, only the Material Class hierarchy was selected to be part of the synonym. However, there is always a flat hierarchy with the same name as the dimension. The flat hierarchy contains a member list of leaf nodes and is always defined as a level hierarchy, so two hierarchies are listed in the [0MATERIAL] dimension.

Dimension:

```
DIMENSION=[0MATERIAL], CAPTION='Material', $
```

Flat hierarchy, same caption as the dimension:

```
HIERARCHY=[0MATERIAL], CAPTION='Material'... $
```

Material Class hierarchy:

```
HIERARCHY=[0MATERIAL                  001], CAPTION='Material class' ...
```

Example: Describing a Level Hierarchy

Each level of the hierarchy is assigned a field name consisting of the hierarchy name (for example, MATERIAL_CLASS) with the level number appended. Each field declaration also specifies the field name of its parent with the WITHIN attribute. The value stored in this field is the member's caption (title).

For the level 1 field MATERIAL_CLASS_LEVEL_01, the parent is the MATERIAL_CLASS dimension.

For the level 2 field MATERIAL_CLASS_LEVEL_02, the parent is MATERIAL_CLASS_LEVEL_01.

If a new level appears in the data, the synonym must be recreated to define this new level. The cardinality for each level is its number of members.

The following field describes level three. Its parent is level 2:

```
$ [0MATERIAL                    001].[LEVEL03] Cardinality 86
 FIELDNAME=MATERIAL_CLASS_LEVEL_03, ALIAS=LEVEL03, USAGE=A40, ACTUAL=A40,
      MISSING=ON,
      TITLE='Material class',
      WITHIN=MATERIAL_CLASS_LEVEL_02,
      PROPERTY=CAPTION,  $
```

The following field represents the flat hierarchy. There are two fields associated with this hierarchy. The first contains the member caption and the second contains the member name:

```
FIELDNAME=MATERIAL_LEVEL_01, ALIAS=LEVEL01, USAGE=A40, ACTUAL=A40,
      MISSING=ON,
      TITLE='Material Member Caption',
      WITHIN='*[0MATERIAL]',
      PROPERTY=CAPTION,  $
    FIELDNAME=MATERIAL_NAME, ALIAS=MEMBER_NAME, USAGE=A18, ACTUAL=A18,
      MISSING=ON,
      TITLE='Material Member Name',
      REFERENCE=MATERIAL_LEVEL_01, PROPERTY=NAME,  $
```

Example: Describing a Parent/Child Hierarchy

Several fields are used to define a parent/child hierarchy. Each has a PROPERTY attribute that describes which hierarchy property it represents. The hierarchy field names are formed by appending a suffix to the hierarchy name.

For example, the caption of a hierarchy named MATERIAL_CLASS is stored in a field whose name is MATERIAL_CLASS_CAPTION and whose property attribute is PROPERTY=CAPTION.

The following table describes the hierarchy fields:

Description of Data	PROPERTY=	Field Suffix
Member Unique ID (unique within the cube)	UID	none
Member Name (unique within the hierarchy)	NAME	_NAME
Member Level Number	LEVEL_NUMBER	_LVLNO
Member Parent	PARENT_OF	_PARENT
Parent Level Number	PARENT_LEVEL_NUMBER	_PARENT_LVLNO
Number of Children	CHILDREN_CARDINALITY	_CHILDREN_CARD
Member Caption (title on reports)	CAPTION	_CAPTION

The following declaration for the MATERIAL_CLASS hierarchy describes the field that contains the unique ID of a member (PROPERTY=UID):

```
FIELDNAME=MATERIAL_CLASS, USAGE=A143, ACTUAL=A143,
  MISSING=ON,
  TITLE='Material class',
  WITHIN='*[0MATERIAL                    001]',
  REFERENCE=[0MATERIAL], PROPERTY=UID,  $
```

The following declaration for the MATERIAL_CLASS hierarchy defines the field that contains a member title (PROPERTY=CAPTION):

```
FIELDNAME=MATERIAL_CLASS_CAPTION, USAGE=A60, ACTUAL=A60,
  MISSING=ON,
  TITLE='Material class CAPTION',
  REFERENCE=MATERIAL_CLASS, PROPERTY=CAPTION,  $
```

Each parent/child hierarchy has three additional fields defined to make reports easier to read:

- The key (field name suffix _KEY). The key is the first token in the member name.
- The key and caption separated by a blank (field name suffix _KEY_CAP).
- The caption and key separated by one blank (field name suffix _CAP_KEY).

For example, for the MATERIAL_CLASS hierarchy, the DEFINE fields are named:

- MATERIAL_CLASS_KEY

- MATERIAL_CLASS _KEY_CAP
- MATERIAL_CLASS_CAP_KEY

Example: Dimension Properties

Following all of a dimension hierarchies, the dimension properties (called *attributes* in SAP BW) are described. Each of these has PROPERTY=UDA (User Defined Attribute) in the synonym. For example the following field represents a member's net weight property:

```
FIELDNAME=NET_WEIGHT__KEY_, ALIAS='20NET_WEIGHT', USAGE=A9, ACTUAL=A9,
      MISSING=ON,
      TITLE='Net weight (Key)',
      REFERENCE=[0MATERIAL], PROPERTY=UDA,  $
```

Example: Sample Request Using a Level Hierarchy

A report request against a level hierarchy must specify the field name for each level of the hierarchy required in the report. For example, the following request displays the sales volume measure for levels one through three of the Material Class hierarchy:

```
TABLE FILE ZLEVEL
SUM SALES_VOLUME
BY MATERIAL_CLASS_LEVEL_01 AS 'LEVEL1'
BY MATERIAL_CLASS_LEVEL_02 AS 'LEVEL2'
BY MATERIAL_CLASS_LEVEL_03 AS 'LEVEL3'
ON TABLE COLUMN-TOTAL
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

LEVEL1	LEVEL2	LEVEL3	Sales Volume
Products	Computer systems	Computer accessories	67,967,525.71
		Computer hardware	17,874,768.15
	Paints / aux. and operating supplies	Paints	.00
	Pumps	Pumps (complete)	59,367,205.93
TOTAL			145,209,499.79

In order to get a total for the entire hierarchy, you have to use an ON TABLE COLUMN-TOTAL command in the request or add another SUM command without a BY phrase (and this would add another column to the report output).

Example: Sample Request Using a Parent/Child Hierarchy

A report request against a parent/child hierarchy can use the BY HIERARCHY phrase to report against the entire hierarchy. The output is automatically formatted with appropriate indentations to show the hierarchy levels and relationships. For example, the following request shows the sales volume measure for three levels of the Material Class hierarchy:

```
TABLE FILE ZOPT
SUM SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
SHOW TO DOWN 3
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

Material class Member Caption	Sales Volume
Material class	145,209,499.79
Products	145,209,499.79
Computer systems	85,842,293.86
Computer hardware	17,874,768.15
Computer accessories	67,967,525.71
Paints / aux. and operating supplies	.00
Paints	.00
Pumps	59,367,205.93
Pumps (complete)	59,367,205.93

The report request does not have to reference specific hierarchy levels. The BY HIERARCHY phrase recreates and formats the hierarchy for display. You can also use a WHEN phrase to select a portion of the hierarchy and a SHOW phrase to specify how many levels above and below the selected portion of the hierarchy you want to display. This display format clearly shows the parent/child relationships between the hierarchy members.

In addition, you can specify whether you want the measure values for each parent to represent aggregates for all of its children (full total) or only those selected for display (visual total).

For more information, see *Hierarchical Reporting* on page 2163.

Hierarchical Reporting

When you issue a request against a cube, some of the requirements and features available depend on the type of hierarchy defined in the synonym.

With a parent/child hierarchy, you can specify whether the measure values displayed for each parent should show the sum of all of its descendants (full total) or the sum of its displayed descendants (visual total). For level hierarchies, the report always displays full totals, which are the values actually stored in the cube.

When a synonym defines parent/child hierarchies, you can use the BY HIERARCHY phrase to sort and format the hierarchy. You can also limit the portion of the hierarchy selected for display using the WHEN phrase.

When a hierarchical request is processed, the first step is to build the hierarchy and mark which nodes should be included, which should be excluded, and which are needed for context.

The next stage fills the hierarchy with measure values. This stage applies WHERE criteria at the leaf nodes to further qualify the members selected for the report. Dimension properties cannot be used in the initial selection phase of the request, but can be used to screen the selected rows based on dimension data.

Measure values also cannot be used to select hierarchy levels for reporting. After the hierarchy rows have been selected, screened, and aggregated, WHERE TOTAL tests can limit the rows displayed based on measure values.

Syntax: How to Display Parent/Child Hierarchies

The following syntax can be used to generate hierarchical reports when the synonym defines parent/child hierarchies:

```
SUM [FROLL.]measure_field ...
BY hierarchy_field [HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO {BOTTOM|DOWN m}] [byoption [WHEN condition] ...] ]
.
.
.
[WHERE expression_using_dimension_data]
.
.
.
[ON hierarchy_field HIERARCHY [WHEN expression_using_hierarchy_fields;]
[SHOW [TOP|UP n] [TO BOTTOM|DOWN m] [byoption [WHEN condition] ...]]
```

where:

FROLL

Specifies a full roll-up of the measure. With a full roll-up, the value displayed is the value found in the cube. This value may not reflect the sum of its displayed descendants if some descendants are eliminated from the output based on the WHEN and SHOW options. When FROLL is not specified, the value displayed is a visual total, which means it is the total of the values for its displayed descendants.

Note: If a request uses WHERE criteria to screen out some data, FROLL will not display the value found in the cube. It will display the roll-up of the selected data.

measure_field

Is the field name of a measure.

BY *hierarchy_field* HIERARCHY

Identifies the hierarchy used for sorting. The field must be a hierarchy field.

ON *hierarchy_field* HIERARCHY

Identifies the hierarchy used for sorting. The field must be a hierarchy field. The request must include either a BY phrase or a BY HIERARCHY phrase for this field name.

WHEN *expression_using_hierarchy_fields*;

Selects hierarchy members. The WHEN phrase must immediately follow the word HIERARCHY to distinguish it from a WHEN phrase associated with a BY option (such as SUBFOOT). Any expression using only hierarchy fields is supported. The WHEN phrase can be on the BY HIERARCHY command or the ON HIERARCHY command, but not both.

SHOW

Specifies which levels to show on the report output relative to the levels selected by the WHEN phrase. If there is no WHEN phrase, the SHOW option is applied to the root node of the hierarchy. The SHOW option can be specified on the BY HIERARCHY phrase or the ON HIERARCHY phrase, but not both.

n

Is the number of ascendants above the set of selected members that will have measure values. All ascendants appear on the report to show the hierarchical context of the selected members. However, ascendants that are not included in the SHOW phrase appear on the report with missing data symbols in the report columns that display measures. The default for *n* is 0.

TOP

Specifies that ascendant levels to the root node of the hierarchy will be populated with measure values.

TO

Is required when specifying a SHOW option for descendant levels.

BOTTOM

Specifies all descendants to the leaf nodes of the hierarchy will be populated with measure values. This is the default value.

m

Is the number of descendants of each selected level that will display. The default for *m* is BOTTOM, which displays all descendants.

byoption

Is one of the following sort-based options: PAGE-BREAK, REPAGE, RECAP, RECOMPUTE, SKIP-LINE, SUBFOOT, SUBHEAD, SUBTOTAL, SUB-TOTAL, SUMMARIZE, UNDER-LINE. If you specify SUBHEAD or SUBFOOT, you must place the WHEN phrase on the line following the heading or footing text.

condition

Is a logical expression.

expression_using_dimension_data

Screens the rows selected in the BY/ON HIERARCHY and WHEN phrases based on dimension data. The expression can use dimension properties and hierarchy fields. However, the selection criteria are always applied to the values at the leaf nodes. Therefore, you cannot use WHERE to select rows based on hierarchy field values that occur at higher levels. For example, in a dimension with Continents, Countries, and Cities, your request will not display any rows if you use WHERE to select a Country name, but it may if you use it to select a City name.

The following examples illustrate hierarchical reporting using the BY HIERARCHY phrase. The zopt Master File was created with parent/child hierarchies by selecting *optimized* as the hierarchy type when creating the synonym, based on the cube 0SD_C01/ZTSCQ31CQ1.

Example: Reporting on a Whole Hierarchy

The following request produces visual totals (SALES_VOLUME) and full totals (FROLL.SALES_VOLUME). The BY HIERARCHY phrase specifies hierarchical reporting. There is no WHEN phrase to limit the portion of the hierarchy displayed. Also note that the full roll-up is displayed with the prefix FSUM added to the field name. The full total is the same as the visual total for a report on the entire hierarchy:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
ON TABLE SET PAGE NOPAGE
ON TABLE SET SCREEN PAPER
ON TABLE SET LINES 88
ON TABLE SUBHEAD
"Reporting on a Whole Hierarchy"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Reporting on a Whole Hierarchy

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	145,209,499.79	145,209,499.79
Products	145,209,499.79	145,209,499.79
Computer systems	85,842,293.86	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxitec-R 375 Personal computer	99,456.50	99,456.50
Maxitec-R 3100 Personal computer	1,848,842.20	1,848,842.20
Maxitec-R 3300 Professional PC	.00	.00
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20	4,681,064.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02	5,008,767.02
Processors	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CD-ROM drives	20,169.00	20,169.00
CD ROM Drive	20,169.00	20,169.00
Computer accessories	67,967,525.71	67,967,525.71
Keyboards	3,285,675.73	3,285,675.73
Standard Keyboard - EURO Model	614,686.79	614,686.79
Standard Keyboard - EURO-Special Model	619,987.95	619,987.95
Professional keyboard - PROFITEC Model	575,433.19	575,433.19
Professional keyboard - MAXITEC Model	689,355.86	689,355.86
Professional keyboard - NATURAL Model	786,211.94	786,211.94
Printer	.00	.00
Other printers	.00	.00
High Speed Printer	.00	.00

Monitors	64,681,849.98	64,681,849.98
Sunny Sunny 01	2,325,600.85	2,325,600.85
Sunny Xa1	2,928,315.00	2,928,315.00
Sunny Tetral3	3,050,854.49	3,050,854.49
Sunny Extreme	3,292,012.09	3,292,012.09
Flatscreen LE 50 P	1,174,441.69	1,174,441.69
Flatscreen MS 1460 P	2,643,840.60	2,643,840.60
Flatscreen LE 64P	2,313,955.17	2,313,955.17
Flatscreen MS 1575P	3,316,991.41	3,316,991.41
Flatscreen MS 1585	3,611,043.69	3,611,043.69
Flatscreen MS 1775P	3,939,392.43	3,939,392.43
Flatscreen MS 1785P	5,042,300.06	5,042,300.06
MAG DX 15F/Fe	2,386,685.30	2,386,685.30
MAG DX 17F	2,688,350.99	2,688,350.99
MAG PA/DX 175	3,059,434.33	3,059,434.33
SEC Multisync XV15	3,129,661.92	3,129,661.92
SEC Multisync XV 17	3,686,004.42	3,686,004.42
Jotachi SN4000	4,908,515.84	4,908,515.84
Jotachi SN4500	2,571,753.78	2,571,753.78
Jotachi SN5000	3,085,110.71	3,085,110.71
Jotachi SN 7000	3,005,405.91	3,005,405.91
TFT Monitor, 17"	871,063.60	871,063.60
PAQ Monitor, 20", Color	1,651,115.70	1,651,115.70
Paints / aux. and operating supplies	.00	.00
Paints	.00	.00
Coating Matt Green RAL 6014/10 Liter	.00	.00
Pumps	59,367,205.93	59,367,205.93
Pumps (complete)	59,367,205.93	59,367,205.93
Pump PRECISION 100	2,964,954.28	2,964,954.28
Pump PRECISION 101	6,407,899.03	6,407,899.03
Pump PRECISION 102	11,213,145.65	11,213,145.65
Pump PRECISION 103	10,406,910.38	10,406,910.38
Pump PRECISION 104	14,328,130.56	14,328,130.56
Pump cast steel IDESNORM 170-230	4,937,452.91	4,937,452.91
Pump standard IDESNORM 100-402	9,108,713.12	9,108,713.12

Example: Selecting a Hierarchy Member

The following request produces a visual total and full total for the MATERIAL_CLASS hierarchy of the MATERIAL dimension, but limits the members selected for display with the WHEN phrase. Note that the hierarchy member selected is displayed in its context (all ancestors to the root of the hierarchy). However, the ancestors are not requested in the report and are, therefore, displayed with missing data symbols. All descendants of the selected members appear in the report output because the default SHOW option for descendants is BOTTOM:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Hierarchy Member"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Hierarchy Member

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	.	.
Products	.	.
Computer systems	.	.
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxitec-R 375 Personal computer	99,456.50	99,456.50
Maxitec-R 3100 Personal computer	1,848,842.20	1,848,842.20
Maxitec-R 3300 Professional PC	.00	.00
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20	4,681,064.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02	5,008,767.02
Processors	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CD-ROM drives	20,169.00	20,169.00
CD ROM Drive	20,169.00	20,169.00

Example: Selecting a Member and Adding a Parent

In the following request, the SHOW option UP 1 TO DOWN 0 added to the WHEN phrase adds the parent (Computer systems) of the selected member (Computer hardware). This parent now contains values for the measures rather than missing data symbols. However, the full total column for the parent contains the sum of all of its descendants, not just the selected Computer hardware member, while the visual total shows the total only for the Computer hardware member:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW UP 1 TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding a Parent"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding a Parent

		FSUM
Material class Member Caption	Sales Volume	Sales Volume
Material class	.	.
Products	.	.
Computer systems	17,874,768.15	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15

Example: **Selecting a Member and Adding Children**

In the following request, the SHOW option UP 0 TO DOWN 1 added to the WHEN phrase adds the children (Computer and Computer components) of the selected member (Computer hardware). Because no children of the selected member are excluded, and no higher level members are in the SHOW set, the full and visual totals are the same:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW UP 0 TO DOWN 1
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding Children"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding Children

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	.	.
Products	.	.
Computer systems	.	.
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Computer components	15,926,469.45	15,926,469.45

Example: Selecting a Member and Showing All Ascendants

In the following request, the SHOW option TOP TO DOWN 0 added to the WHEN phrase adds all ascendants but no descendants of the selected member (Computer hardware). These parents now contain values for the measures rather than missing data symbols. However, the full total column for the ascendants contains the sum of all of their descendants, not just the specified Computer hardware member, while the visual total shows the total for only the Computer hardware member:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW TOP TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding All Ascendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding All Ascendants

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	17,874,768.15	145,209,499.79
Products	17,874,768.15	145,209,499.79
Computer systems	17,874,768.15	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15

Example: Selecting a Member and Showing All Descendants

In the following request, the absence of a SHOW phrase is equivalent to SHOW UP 0 TO BOTTOM. The WHEN phrase selects the member Computer hardware. Because no children of the selected member are excluded and no higher level members are in the SHOW set, the full and visual totals are the same, and the ascendant levels display missing data symbols:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Showing All Descendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Showing All Descendants

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	.	.
Products	.	.
Computer systems	.	.
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxitec-R 375 Personal computer	99,456.50	99,456.50
Maxitec-R 3100 Personal computer	1,848,842.20	1,848,842.20
Maxitec-R 3300 Professional PC	.00	.00
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20	4,681,064.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02	5,008,767.02
Processors	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CD-ROM drives	20,169.00	20,169.00
CD ROM Drive	20,169.00	20,169.00

Example: Selecting a Member and Showing All Ascendants and Descendants

In the following request, the SHOW option TOP added to the WHEN phrase adds all ascendants and descendants (since TO BOTTOM is the default) of the selected member (Computer hardware). These parents are now in the SHOW set and contain values for the measures rather than missing data symbols. However, the full total column for the ascendants contains the sums of all of their descendants, not just the specified Computer hardware member, while the visual total shows the total for only the Computer hardware member:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW TOP
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting a Member and Adding All Ascendants and Descendants"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting a Member and Adding All Ascendants and Descendants

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	17,874,768.15	145,209,499.79
Products	17,874,768.15	145,209,499.79
Computer systems	17,874,768.15	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxitec-R 375 Personal computer	99,456.50	99,456.50
Maxitec-R 3100 Personal computer	1,848,842.20	1,848,842.20
Maxitec-R 3300 Professional PC	.00	.00
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20	4,681,064.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02	5,008,767.02
Processors	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CD-ROM drives	20,169.00	20,169.00
CD ROM Drive	20,169.00	20,169.00

Example: Displaying Members of a Range of Hierarchy Levels

The following request uses a WHEN phrase on the MATERIAL_CLASS_LVLNO field to display members of levels 0 through 3 of the hierarchy. The SHOW option TO DOWN 0 eliminates printing of any descendant levels not selected by the WHEN phrase. The full and visual totals are the same because no children were excluded when calculating the visual totals:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_LVLNO LE 3;
SHOW TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Displaying Members of a Range of Hierarchy Levels"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Displaying Members of a Range of Hierarchy Levels

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	145,209,499.79	145,209,499.79
Products	145,209,499.79	145,209,499.79
Computer systems	85,842,293.86	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15
Computer accessories	67,967,525.71	67,967,525.71
Paints / aux. and operating supplies	.00	.00
Paints	.00	.00
Pumps	59,367,205.93	59,367,205.93
Pumps (complete)	59,367,205.93	59,367,205.93

Example: Selecting Members for Display and Screening on Member Values

You use the WHEN phrase to select hierarchy members to display on the report. Each of the selected members is displayed in its context. The ascendants of selected members display missing data symbols. When a request is processed, the first step is to build the hierarchy and mark which nodes should be included, which should be excluded, and which are needed for context but should display missing data symbols according to the WHEN phrase.

The next stage fills the hierarchy with measure values. This stage applies WHERE criteria (which must select on the lowest level of the hierarchy or the report will be empty).

If a WHERE is specified without a WHEN, no nodes are marked as excluded, so no nodes display missing values.

The following request selects the portion of the hierarchy whose members contain the values Computer systems or Pumps. It shows the parents of these members and all levels of descendants. Note that the dimension property MATERIAL_TYPE__MEDIUM_NAME_ is displayed in the request and that only the leaf members have values for this property:

```
TABLE FILE ZOPT
WRITE MATERIAL_TYPE__MEDIUM_NAME_ AS 'MEDIUM NAME' SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer systems' OR 'Pumps';
SHOW UP 1
ON TABLE SET PAGE NOPAGE
ON TABLE SET SCREEN PAPER
ON TABLE SET LINES 88
ON TABLE SUBHEAD
"Selecting Hierarchy Members"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting Hierarchy Members

Material class Member Caption	MEDIUM NAME	Sales Volume
Material class	.	.
Product		145,209,499.79
Computer systems		85,842,293.86
Computer hardware		17,874,748.15
Computer		1,948,298.70
Desktop computer		1,948,298.70
Maxitec-R.375 Personal computer	Finished product	99,456.50
Maxitec-R.3100 Personal computer	Finished product	1,848,842.20
Maxitec-R.3300 Professional PC	Finished product	.00
Computer components		15,926,449.45
RAM		545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	Trading goods	545,330.61
Hard disk		13,845,934.28
Harddisk 10.80 GB / SCSI-2-Fast	Trading goods	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	Trading goods	4,681,044.20
Harddisk 42.94 GB / SCSI-2-Fast	Trading goods	5,008,787.02
Processor		1,495,035.56
Processor 700 MHz	Trading goods	1,072,938.99
3Processor 500 MHz	Trading goods	422,096.57
Drives		20,169.00
CD-ROM drives		20,169.00
CD ROM Drive	Trading goods	20,169.00
Computer accessories		67,947,525.71
Keyboards		3,285,675.73
Standard Keyboard - EURO Modal	Trading goods	614,686.79
Standard Keyboard - EURO Special Modal	Trading goods	619,987.95
Professional keyboard - PROFITEC Modal	Trading goods	575,433.19
Professional keyboard - MAXITEC Modal	Trading goods	689,355.86
Professional keyboard - NATURAL Modal	Trading goods	786,211.94
Printer		.00
Other printers		.00
High Speed Printer	Trading goods	.00

Monitors		64,681,849.98
Sunny Sunny 01	Trading goods	2,325,400.85
Sunny Xal	Trading goods	2,928,315.00
Sunny IntraB	Trading goods	3,050,854.49
Sunny Extrema	Trading goods	3,292,012.09
Flatcman LE 50 P	Trading goods	1,174,441.69
Flatcman MS 1440 P	Trading goods	2,443,840.60
Flatcman LE 44P	Trading goods	2,313,955.17
Flatcman MS 1575P	Trading goods	3,316,991.41
Flatcman MS 1585	Trading goods	3,411,043.69
Flatcman MS 1775P	Trading goods	3,939,392.43
Flatcman MS 1785P	Trading goods	5,042,300.06
MAG DX 15WP6	Trading goods	2,386,685.30
MAG DX 17F	Trading goods	2,488,350.99
MAG PAJDX 175	Trading goods	3,039,434.33
SEC Multisync XV15	Trading goods	3,129,661.92
SEC Multisync XV17	Trading goods	3,484,004.42
Jotachi SN4000	Trading goods	4,908,515.84
Jotachi SN4500	Trading goods	2,571,753.78
Jotachi SN5000	Trading goods	3,085,110.71
Jotachi SN 7000	Trading goods	3,005,405.91
IFI Monitor, 17"	Trading goods	871,043.60
PAQ Monitor, 20", Color	Trading goods	1,451,115.70
Pumps		59,347,205.98
Pumps (complete)		59,347,205.98
Pump PRECISION 100	Finished product	2,944,954.28
Pump PRECISION 101	Finished product	6,407,899.03
Pump PRECISION 102	Finished product	11,213,145.45
Pump PRECISION 103	Finished product	10,404,910.38
Pump PRECISION 104	Finished product	14,328,130.56
Pump cast steel IDE SNORM 170-230	Finished product	4,937,452.91
Pump standard IDE SNORM 100-402	Finished product	9,108,713.12

The following request selects the portion of the hierarchy whose members contain the values Computer systems or Pumps. It then screens on the dimension property MATERIAL_TYPE__MEDIUM_NAME_ equal to Finished product. This WHERE screening condition does not affect the portion of the hierarchy that was selected by WHEN, but it does not display or include in the totals those rows that had no data meeting the WHERE condition:

```
TABLE FILE ZOPT
WRITE MATERIAL_TYPE__MEDIUM_NAME_ AS 'MEDIUM NAME' SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer systems' OR 'Pumps';
SHOW UP 1
WHERE MATERIAL_TYPE__MEDIUM_NAME_ EQ 'Finished product';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Selecting Members and Screening Data Values"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Selecting Members and Screening Data Values

Material class Member Caption	MEDIUM NAME	Sales Volume
Material class	.	.
Products		61,315,504.63
Computer systems		1,948,298.70
Computer hardware		1,948,298.70
Computer		1,948,298.70
Desktop computer		1,948,298.70
Maxitec-R 375 Personal computer	Finished product	99,456.50
Maxitec-R 3100 Personal computer	Finished product	1,848,842.20
Maxitec-R 3300 Professional PC	Finished product	.00
Pumps		59,367,205.93
Pumps (complete)		59,367,205.93
Pump PRECISION 100	Finished product	2,964,954.28
Pump PRECISION 101	Finished product	6,407,899.03
Pump PRECISION 102	Finished product	11,213,145.65
Pump PRECISION 103	Finished product	10,406,910.38
Pump PRECISION 104	Finished product	14,328,130.56
Pump cast steel IDESNORM 170-230	Finished product	4,937,452.91
Pump standard IDESNORM 100-402	Finished product	9,108,713.12

Example: Comparing Member Selection With Data Screening

The following request selects hierarchy levels less than or equal to 3 and screens values at the lowest level of the hierarchy for captions containing Flatscreen or Harddisk. The SHOW option TO DOWN 0 prevents the display of descendants of the selected members:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_LVLNO LE 3;
SHOW TO DOWN 0
WHERE MATERIAL_CLASS_CAPTION CONTAINS 'Flatscreen' OR 'Harddisk';
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Comparing Member Selection With Data Screening"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Comparing Member Selection With Data Screening

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	35,907,899.33	35,907,899.33
Products	35,907,899.33	35,907,899.33
Computer systems	35,907,899.33	35,907,899.33
Computer hardware	13,865,934.28	13,865,934.28
Computer accessories	22,041,965.05	22,041,965.05

Example: **Using SHOW Without WHEN**

The following request does not contain a WHEN phrase, so the SHOW option DOWN TO 0 applies to the root node (as if there had been a WHEN phrase that selected only the root node). Therefore, the root level is the only one shown on the report output:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
SHOW TO DOWN 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"SHOW Without WHEN"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

SHOW Without WHEN

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	145,209,499.79	145,209,499.79

Example: Using SKIP-LINE

The following request adds the BY option SKIP-LINE to the BY HIERARCHY phrase:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW UP 2 TO DOWN 2 SKIP-LINE
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using SKIP-LINE"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using SKIP-LINE

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	.	.
Products	17,874,768.15	145,209,499.79
Computer systems	17,874,768.15	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Processors	1,495,035.56	1,495,035.56
Drives	20,169.00	20,169.00

Example: Using Sort Options Conditional on a Measure

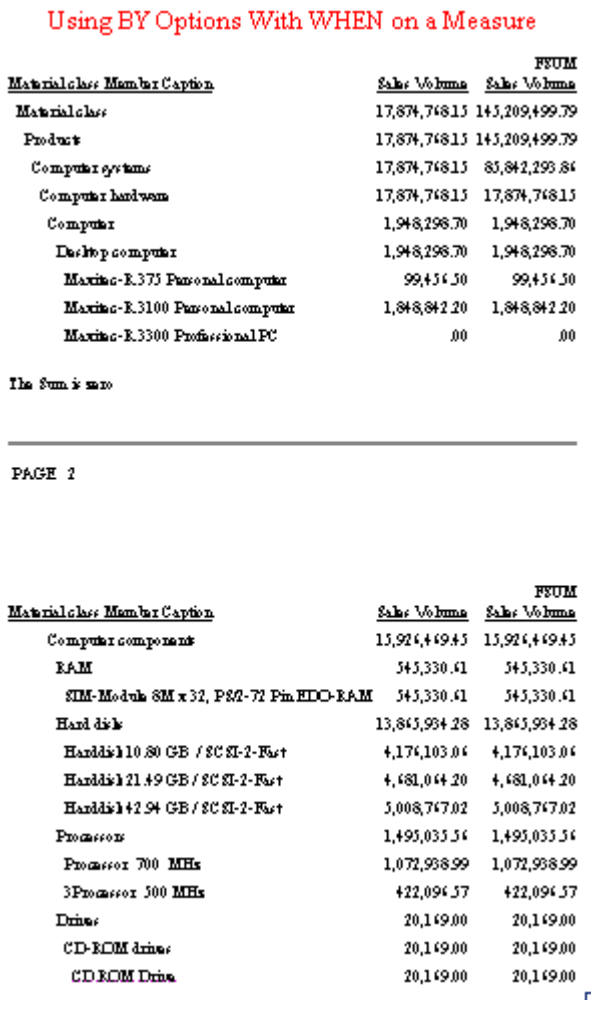
The following request has a BY HIERARCHY command with a WHEN phrase to select members as well as a WHEN phrase to control the UNDER-LINE option.

The SUBFOOT and PAGE-BREAK options are in two ON phrases that reference the same hierarchy field (all of the BY options could have been on the BY HIERARCHY phrase).

Each BY option has its own WHEN phrase. Note that the WHEN phrases for the BY options use a measure field in their expressions. In this example, all of the BY options are activated when the sales volume value is zero:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW TOP UNDER-LINE WHEN SALES_VOLUME EQ 0;
ON MATERIAL_CLASS_CAPTION SUBFOOT
" "
"The Sum is zero"
" "
WHEN SALES_VOLUME EQ 0;
ON MATERIAL_CLASS_CAPTION PAGE-BREAK WHEN SALES_VOLUME EQ 0;
ON TABLE SUBHEAD
"Using BY Options With WHEN on a Measure"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:



Using BY Options With WHEN on a Measure

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	17,874,748.15	145,209,499.79
Product	17,874,748.15	145,209,499.79
Computer systems	17,874,748.15	85,842,293.86
Computer hardware	17,874,748.15	17,874,748.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxitec-R.375 Personal computer	99,456.50	99,456.50
Maxitec-R.3100 Personal computer	1,848,842.20	1,848,842.20
Maxitec-R.3300 Professional PC	.00	.00

The Sum is zero

PAGE 2

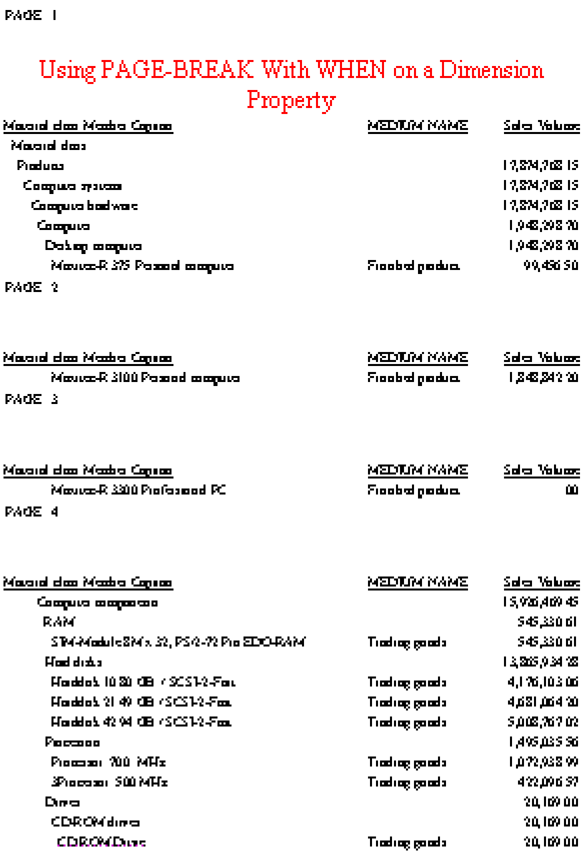
Material class Member Caption	Sales Volume	FSUM Sales Volume
Computer components	15,926,449.45	15,926,449.45
RAM	545,330.61	545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disk	13,845,934.28	13,845,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,044.20	4,681,044.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,787.02	5,008,787.02
Processor	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CD-ROM drives	20,169.00	20,169.00
CD ROM Drive	20,169.00	20,169.00

Example: Using Sort Options Conditional on a Dimension Property

The following request has a BY HIERARCHY command with a WHEN phrase to select members. The ON phrase has a PAGE-BREAK option that is activated when MATERIAL_TYPE__MEDIUM_NAME_ equals 'Finished product':

```
TABLE FILE ZOPT
WRITE MATERIAL_TYPE__MEDIUM_NAME_ AS 'MEDIUM NAME'
 SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW UP 2
ON MATERIAL_CLASS_CAPTION PAGE-BREAK
WHEN MATERIAL_TYPE__MEDIUM_NAME_ EQ 'Finished product';
ON TABLE SUBHEAD
"Using PAGE-BREAK With WHEN on a Dimension Property"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:



Example: Using Sort Options Conditional on a Hierarchy Field

The following request has a BY HIERARCHY command with a WHEN phrase to select members. The ON phrase has a PAGE-BREAK option that is activated when MATERIAL_CLASS_CAPTION contains 'Harddisk':

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW UP 2
ON MATERIAL_CLASS_CAPTION PAGE-BREAK
WHEN MATERIAL_CLASS_CAPTION CONTAINS 'Harddisk';
ON TABLE SUBHEAD
"Using PAGE-BREAK With WHEN on a Hierarchy Field"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

PAGE 1

Using PAGE-BREAK With WHEN on a Hierarchy Field

Material class Material Caption	Sales Volume	FSUM Sales Volume
Material class		
Product	17,874,768.15	145,209,499.79
Computer system	17,874,768.15	85,342,293.86
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxtor-R 375 Personal computer	99,456.50	99,456.50
Maxtor-R 3100 Personal computer	1,848,842.20	1,848,842.20
Maxtor-R 3300 Professional PC	.00	.00
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
SIM-Module 8MB, 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Harddisk 10.20 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06

PAGE 2

Material class Material Caption	Sales Volume	FSUM Sales Volume
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20	4,681,064.20

PAGE 3

Material class Material Caption	Sales Volume	FSUM Sales Volume
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02	5,008,767.02

PAGE 4

Material class Material Caption	Sales Volume	FSUM Sales Volume
Processor	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CDROM drive	20,169.00	20,169.00
CDROM Drive	20,169.00	20,169.00

Example: Using WHERE TOTAL to Screen on a Measure

The following request has a BY HIERARCHY phrase with a WHEN phrase to select members. The WHERE TOTAL phrase selects rows in which the sales volume is not zero:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME FROLL.SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware';
SHOW TOP
WHERE TOTAL SALES_VOLUME NE 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using WHERE TOTAL to Screen on a Measure"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using WHERE TOTAL to Screen on a Measure

Material class Member Caption	Sales Volume	FSUM Sales Volume
Material class	17,874,768.15	145,209,499.79
Products	17,874,768.15	145,209,499.79
Computer systems	17,874,768.15	85,842,293.86
Computer hardware	17,874,768.15	17,874,768.15
Computer	1,948,298.70	1,948,298.70
Desktop computer	1,948,298.70	1,948,298.70
Maxitec-R 375 Personal computer	99,456.50	99,456.50
Maxitec-R 3100 Personal computer	1,848,842.20	1,848,842.20
Computer components	15,926,469.45	15,926,469.45
RAM	545,330.61	545,330.61
SIM-Module 8M x 32, PS/2-72 Pin EDO-RAM	545,330.61	545,330.61
Hard disks	13,865,934.28	13,865,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20	4,681,064.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02	5,008,767.02
Processors	1,495,035.56	1,495,035.56
Processor 700 MHz	1,072,938.99	1,072,938.99
3Processor 500 MHz	422,096.57	422,096.57
Drives	20,169.00	20,169.00
CD-ROM drives	20,169.00	20,169.00
CD ROM Drive	20,169.00	20,169.00

Example: Using Two BY HIERARCHY Phrases

The following request has a BY HIERARCHY phrase for the Material Class hierarchy and a second BY HIERARCHY phrase for the World/Continents/Countries hierarchy (which is from a different dimension, as required). All selected members for the World/Continents/Countries hierarchy are repeated for each selected member of the Material Class hierarchy. The WHERE TOTAL phrase omits rows in which the sales volume is not zero:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
SHOW TO DOWN 2 AS 'Material Class'
BY WORLD___CONTINENTS___COUNTRIES_CAPTION HIERARCHY
WHEN WORLD___CONTINENTS___COUNTRIES_CAPTION OMITS 'TEST'
AND WORLD___CONTINENTS___COUNTRIES_LVLNO LE 2;
SHOW UP 0 TO DOWN 0 AS 'Region'
WHERE TOTAL SALES_VOLUME NE 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using Two BY HIERARCHY Phrases"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using Two BY HIERARCHY Phrases

Material Class	Region	Sales Volume
Material class	World /Continent	16,591,918.04
	Europe	16,582,667.78
	Norway	6,765,800.00
	Italy	3,871,450.00
	Great Britain	4,881,550.00
	France	1,063,867.78
	North America	9,250.26
	US	9,250.26
Product	World /Continent	16,591,918.04
	Europe	16,582,667.78
	Norway	6,765,800.00
	Italy	3,871,450.00
	Great Britain	4,881,550.00
	France	1,063,867.78
	North America	9,250.26
	US	9,250.26
Computer systems	World /Continent	1,065,516.04
	Europe	1,063,867.78
	France	1,063,867.78
	North America	1,648.26
	US	1,648.26
Pumps	World /Continent	15,526,402.00
	Europe	15,518,800.00
	Norway	6,765,800.00
	Italy	3,871,450.00
	Great Britain	4,881,550.00
	North America	7,602.00

Example: Using BY HIERARCHY and BY Phrases

The following request has a BY phrase for the World/Continents/Countries hierarchy and a BY HIERARCHY phrase for the Material Class hierarchy. All selected members for the Material Class hierarchy are repeated for each selected member of the World/Continents/Countries hierarchy (which is not displayed with hierarchical indentations when referenced in a BY phrase). A BY on a unique field is required. Because the captions are not unique for the World/Continents/Countries hierarchy, there is one BY on a unique field with the NOPRINT option and another BY on the caption field:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME
BY WORLD___CONTINENTS___COUNTRIES NOPRINT
BY WORLD___CONTINENTS___COUNTRIES_CAPTION
BY MATERIAL_CLASS_CAPTION HIERARCHY
SHOW TO DOWN 2
WHERE WORLD___CONTINENTS___COUNTRIES_CAPTION OMITS 'TEST'
AND WORLD___CONTINENTS___COUNTRIES_LVLNO EQ 2;
WHERE TOTAL SALES_VOLUME NE 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using BY HIERARCHY and BY Phrases"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using BY and BY HIERARCHY Phrases

World / Continents / Countries Member Caption	Material class Member Caption	Sales Volume
France	Material class	1,063,867.78
	Products	1,063,867.78
	Computer systems	1,063,867.78
Great Britain	Material class	4,881,550.00
	Products	4,881,550.00
	Pumps	4,881,550.00
Italy	Material class	3,871,450.00
	Products	3,871,450.00
	Pumps	3,871,450.00
Norway	Material class	6,765,800.00
	Products	6,765,800.00
	Pumps	6,765,800.00
US	Material class	9,250.26
	Products	9,250.26
	Computer systems	1,648.26
	Pumps	7,602.00

The following is the same request with the BY HIERARCHY phrase first and the BY phrase second. All selected members for the World/Continents/Countries hierarchy are repeated for each selected member of the Material Class hierarchy:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
SHOW TO DOWN 2
BY WORLD___CONTINENTS___COUNTRIES NOPRINT
BY WORLD___CONTINENTS___COUNTRIES_CAPTION
WHERE WORLD___CONTINENTS___COUNTRIES_CAPTION OMITS 'TEST'
AND WORLD___CONTINENTS___COUNTRIES_LVLNO EQ 2;
WHERE TOTAL SALES_VOLUME NE 0
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using BY HIERARCHY and BY Phrases"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using BY HIERARCHY and BY Phrases

Material class Member Caption	World / Continents / Countries Member Caption	Sales Volume
Material class	France	1,063,867.78
	Great Britain	4,881,550.00
	Italy	3,871,450.00
	Norway	6,765,800.00
	US	9,250.26
Products	France	1,063,867.78
	Great Britain	4,881,550.00
	Italy	3,871,450.00
	Norway	6,765,800.00
	US	9,250.26
Computer systems	France	1,063,867.78
	US	1,648.26
Pumps	Great Britain	4,881,550.00
	Italy	3,871,450.00
	Norway	6,765,800.00
	US	7,602.00

Example: **Using ON HIERARCHY Without BY HIERARCHY**

The following request has a BY phrase and an ON HIERARCHY phrase for the Material Class hierarchy. All hierarchy options and BY options are supported on the ON HIERARCHY phrase. This request also has a WHERE clause that selects rows based on the value of a hierarchy field (MATERIAL_CLASS_CAPTION). The WHERE test selects rows based on values at the leaf nodes (Harddisk is a value found on leaf nodes):

```
TABLE FILE ZOPT
WRITE SALES_VOLUME
BY MATERIAL_CLASS_CAPTION
ON MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION CONTAINS 'Computer';
WHERE MATERIAL_CLASS_CAPTION CONTAINS 'Harddisk'
ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using BY and ON HIERARCHY"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The output is:

Using BY and ON HIERARCHY

Material class Member Caption	Sales Volume
Material class	.
Products	.
Computer systems	13,865,934.28
Computer hardware	13,865,934.28
Computer components	13,865,934.28
Hard disks	13,865,934.28
Harddisk 10.80 GB / SCSI-2-Fast	4,176,103.06
Harddisk 21.49 GB / SCSI-2-Fast	4,681,064.20
Harddisk 42.94 GB / SCSI-2-Fast	5,008,767.02

Reference: **Hierarchical Reports With Multiple Display Commands**

In a request with more than one display command, each command must repeat all of the sort phrases from the previous command in the same order after which additional sort phrases can be added. In a hierarchical reporting request, the following rules must be followed:

- All display commands must be summation commands. PRINT is not supported in a hierarchical reporting request.

- Both BY and BY HIERARCHY sort phrases can be used. As with a non-hierarchical request, all of the sort phrases must be repeated in the same order with a subsequent display command.
- Only one WHEN and one SHOW phrase can be specified for each hierarchy referenced in the request. These phrases can be specified at any level of the request, but they will apply to every display command in the request.

Example: Using Multiple Display Commands in a Hierarchical Report

The following request has two WRITE commands. The first WRITE command has a BY HIERARCHY sort phrase for the MATERIAL CLASS hierarchy and a BY HIERARCHY phrase for the WORLD_CONTINENTS_COUNTRIES hierarchy. The second WRITE command repeats these phrases and adds a BY phrase for the SALESORG dimension:

```
TABLE FILE ZOPT
WRITE SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
WHEN MATERIAL_CLASS_CAPTION EQ 'Computer hardware'; AS 'Material Class'
SHOW TO DOWN 2

BY WORLD___CONTINENTS___COUNTRIES_CAPTION HIERARCHY
WHEN WORLD___CONTINENTS___COUNTRIES_CAPTION OMITS 'TEST'
AND WORLD___CONTINENTS___COUNTRIES_LVLNO EQ 2; AS 'Region'

WRITE SALES_VOLUME SALES_ORDER_ITEM
BY MATERIAL_CLASS_CAPTION HIERARCHY
BY WORLD___CONTINENTS___COUNTRIES_CAPTION HIERARCHY
BY SALES_ORGANIZATION_LEVEL_01 AS 'Sales Organization'

ON TABLE SET PAGE NOPAGE
ON TABLE SUBHEAD
"Using Multiple Display Commands"
ON TABLE SET STYLE *
TYPE=REPORT, SIZE=8,$
TYPE=REPORT, GRID=OFF, WRAP=OFF, $
TYPE=TABHEADING, SIZE=10, COLOR=RED, JUSTIFY=CENTER, $
END
```

The BY HIERARCHY phrase for the MATERIAL CLASS hierarchy has a WHEN phrase that selects members whose caption is *Computer hardware*. The SHOW option (TO DOWN 2) displays two levels of descendants below the Computer hardware level. Ascendants of the selected members display for context, but with missing data symbols instead of values.

For the WORLD_CONTINENTS_COUNTRIES hierarchy, the selected members have level number two and do not have the characters *TEST* in their captions. The SHOW option displays two levels of descendants on the report output. Ascendants display for context, with missing data symbols. All selected members of the WORLD_CLASS_COUNTRIES hierarchy display for each selected member of the MATERIAL CLASS hierarchy.

The output is:

Using Multiple Display Commands

Material Class	Region	Sales Volume	Sales Organization	Sales Volume	Sales Order Item
Material class	.	.	.	.	.
Products	.	.	.	.	.
Computer systems	.	.	.	.	.
Computer hardware	World / Continents	.	.	.	.
	Europe	.	.	.	.
Computer hardware	France	1,061,431.86	.	1,061,431.86	1.00
	IDES France SA	7,925.02	.	7,925.02	.00
	Adecom SA	1,053,506.84	.	1,053,506.84	1.00
Computer	World / Continents	.	.	.	.
	Europe	.	.	.	.
Computer	France	1,061,431.86	.	1,061,431.86	1.00
	IDES France SA	7,925.02	.	7,925.02	.00
	Adecom SA	1,053,506.84	.	1,053,506.84	1.00
Desktop computer	World / Continents	.	.	.	.
	Europe	.	.	.	.
Desktop computer	France	1,061,431.86	France, Paris	1,053,506.84	1.00
			Germany Frankfurt	7,925.02	.00
	IDES France SA	7,925.02	Germany Frankfurt	7,925.02	.00
	Adecom SA	1,053,506.84	France, Paris	1,053,506.84	1.00

Columnar Reporting

If you run a standard TABLE or FML report against a cube, you are running what is known as a Slice report. This two-dimensional report resembles a slice of a larger multi-dimensional cube. A Slice report flattens out the multi-dimensional structure of a cube to show information from one angle.

To create a valid Slice report, you must understand the cube data structure and the logic and assumptions that the Adapter for SAP BW uses when slicing a cube.

Reference: Valid and Invalid Slices-Cube Slicing Logic

The Adapter for SAP BW has built-in logic that enables it to determine the rows and cells to extract from rollups in order to deliver Slice reports. TABLE uses the lowest level BY phrase for each dimension to determine at which rollup to locate the lookup for aggregations. It then rolls the result up from there, performing aggregations on the data as needed to ensure the consistency of the results.

If you do not specify a BY field in your TABLE request, you may receive invalid data or a message. This happens because without a BY field to determine the level of rollup at which to perform the data lookup, TABLE does not know where to source the result.

Example: Displaying Single Dimensions in a Slice Report

```
TABLE FILE MYCUBE
SUM STORE_COST STORE_SALES
BY COUNTRY_LEVEL_01 ON TABLE COLUMN_TOTAL
END
```

This report displays aggregate data sorted against one dimension in the source. The standard measures such as STORE_COST and STORE_SALES are referenced with SUM. Column totals are requested in the report.

Example: Displaying Multiple Dimensions in a Slice Report

```
TABLE FILE MYCUBE
SUM STORE_COST STORE_SALES
PRINT PROFIT
BY COUNTRY_LEVEL_01 BY STATE_PROVINCE_LEVEL BY CITY_LEVEL_01
END
```

This report displays the same data as the previous one, except that it sorts against both dimension hierarchies in the cube. The order of the sort fields follows the logical order of the dimension hierarchy.

Reporting With Variables

A BW query uses variables to supply values at execution time. A variable can have an Entry_Type of either Mandatory/Required or Optional. All mandatory variable values must be supplied in the TABLE or SQL request. A WHERE or IF statement in the request provides the run-time value(s). For example, if the variable COMPANY_CODE is mandatory, it will supply values for company code 430 at run time:

```
WHERE COMPANY_CODE_N EQ '430'
```

All variables passed to BW have a format of 124 alpha bytes. The Master File synonym breaks variables into two parts: variable_C and variable_N.

- The caption (_C) is the first 60 bytes and is an internal reference for BW.
- The name (_N) is the last 64 bytes in which the actual key value will be sent to the query.

SAP BW expects the literals in SAP VARIABLES clauses to contain only the uncompounded part of the member name. Although the Adapter for SAP BW communicates with the user in terms of member names and captions (for variables of MEMBER type), it automatically detects compounding and generates a literal based on the uncompounded part of the supplied member name. In order to obtain meaningful reports when using compounded variables the user has to be familiar with the SAP OSS Note 605208 and follow its directions. Effectively it mandates either creation of a separate variable on the compounding characteristic or a single value hard filter on it. Please see the OSS Note 605208 for further information.

In the previous example, the name (_N) supplies the variable value. You can use either the caption or the name for screening, but you must supply appropriate values for each:

```
company_code_n : WHERE COMPANY_CODE_N EQ 430
company_code_c : WHERE COMPANY_CODE_N EQ 'IDES USA'
```

Tip: If the associated dimension of this variable is available in the query, do not build the selection on that field, but, rather, use the name. In other words, WHERE COMPANY_CODE_LEVEL_01 EQ 430 *will not* populate the variable, while COMPANY_CODE_N will. Additional non-variable selections are available, but they are less efficient. The non-variable conditions are applied after the BW answer set has been extracted.

Variables cannot be displayed in a request. They should be used only for applying selection criteria. You can display the dimensions or measures with which they are associated.

You cannot display a variable as a BY field. Variables can have a selection type of single, interval, or complex.

- Single type variables must have only one value supplied in the selection condition.
- Interval variables can have a single value or a range of values.
- Complex variables have no restrictions.

To supply a range, use two WHERE statements or combine them into one statement using AND. For example, this report needs to specify two years:

```
WHERE FISCAL_YEAR_N EQ '2001'
        OR FISCAL_YEAR_N EQ '2002'
WHERE FISCAL_YEAR_N GE '2001'
WHERE FISCAL_YEAR_N LE '2002'
WHERE FISCAL_YEAR_N GE '2001'
      AND FISCAL_YEAR_N LE '2002'
WHERE FISCAL_YEAR_N IN ('2001', '2002')
WHERE FISCAL_YEAR_N FROM '2001' TO '2002'
```

Selections that use an OR connector, like the first sample above, can only test against the same field or related hierarchy. You cannot combine variable selection and non-variable selection in the same WHERE statement (screening predicate).

Full and Partial Aggregation

Cube requests are requests that reference at least one measure and possibly more than one hierarchy. Variables may also be referenced.

For cube requests, the adapter reads cells with fully or partially aggregated data from the cube and applies certain restrictions on the contents of the TABLE request to ensure consistency of reports.

There are two modes in which cube requests are processed: *Full Aggregation* and *Partial Aggregation*.

- In full aggregation mode, the adapter retrieves aggregated values from the cube exactly as they will be displayed on the report.
- In partial aggregation mode the Adapter retrieves partially aggregated data which is then further aggregated by WebFOCUS.

Full aggregation mode is more efficient because no aggregation is needed beyond what is already stored in the cube.

Reference: Support for Full Aggregation Mode

The following table describes support for full aggregation mode:

Command	Dimension Properties	Measures	Variables
WRITE/ SUM/ADD	Supported, converted to FST.	Supported, converted to an operation that matches the measure aggregator.	Not supported.

Command	Dimension Properties	Measures	Variables
COMPUTE	Supported.	Supported.	Not supported.
PRINT	Supported.	Supported for compatibility. Not supported with BY HIERARCHY.	Not supported.
WHERE TOTAL	Supported.	Supported.	Not supported.
WHERE	Supported.	Not supported.	Required for mandatory variables; must provide required restriction for its variable selection type. Restricted to a group of fields that represent a single variable.
WHEN	Supported.	Not supported.	Not supported.
DEFINE	Supported.	Not supported.	Not supported.
BY	Supported. Requires BY on a unique field of a parent/ child hierarchy (except when referenced only in a WHERE). BY HIERARCHY adds BY internally.	Not supported.	Not supported.
BY HIERARCHY	Supported.	Not supported.	Not supported.

Command	Dimension Properties	Measures	Variables
ON HIERARCHY	Supported. Requires a corresponding BY or BY HIERARCHY phrase.	Not supported.	Not supported.

Reference: **Support for Partial Aggregation Mode**

In partial aggregation mode, there are additional restrictions on the use of measures. Note that in requests with multiple display commands, the partial aggregation restrictions apply to all display commands except for the ones associated with the lowest BY phrase:

- Additive aggregators (those that can be further aggregated from partial totals such as SUM., CNT., MAX., and MIN.) are supported as long as they match the measure aggregator.
- Non-additive aggregators (such as AVE.) are not supported.

The following factors turn on partial aggregation mode:

- Multiple display commands.
- Parent/Child hierarchies: Visual totals or Full totals with a WHERE on the hierarchy.
- Level hierarchies: Not all levels of a hierarchy that are referenced in the request have BY phrases.
- All hierarchies: The hierarchy is referenced only in a WHERE clause that selects more than one member.

Support for Time Dependent Hierarchies

The contents of a hierarchy may change over time. As changes occur, multiple versions of the hierarchy for different time periods accumulate. Results of a report from the cube depend on which of the hierarchy versions is selected at report time.

A cube can be created that has multiple time-dependent hierarchies, each having different versions for different time periods.

When you create a report against such a cube, you cannot arbitrarily choose the versions of time-dependent hierarchies because only certain combinations of versions make business sense. The way to choose versions of time-dependent hierarchies is by indicating what is called the *key date*. When you specify a key date you are saying you want to see the contents of the cube as of that effective date. The BW system automatically chooses the proper combination of versions of the time-dependent hierarchies valid for this key date.

Time-dependent hierarchies sometimes may also have multiple concurrent versions of the same hierarchy. The number of these versions changes with time and is reflected in the cube metadata. Depending on the key date, the BW system exposes a varying number of these concurrent versions, each as a separate distinctly named hierarchy. Thus, not only the contents of a hierarchy can change with time, but also the number of its concurrent (simultaneously existing) versions can change.

When you create a synonym against a cube with time dependent hierarchies, you can specify a key date and produce a Master File that contains all versions of the hierarchies (time dependent and not) that existed for the time period associated with the specified key date. In the synonym, different concurrent versions of the same hierarchy have different field names that let you distinguish between the versions. Specifying the key date at the time of synonym creation makes sense mostly in the presence of a time-dependent hierarchy that has or has had multiple concurrent versions. The key date should be specified for a period of time when the hierarchy had the maximum number of concurrent versions.

If you do not specify a key date when creating a synonym, the current date is used as the key date. If all the time-dependent hierarchies in the cube have only time-changing contents but do not have varying numbers of concurrent versions, specification of the key date at create synonym time makes sense only when simultaneously requesting member sampling. Otherwise, the resulting synonym will be identical no matter what key date has been specified.

When you report against a time-dependent hierarchy, you use the ENGINE BWBAPI SET KEY_DATE command to set a key date for subsequent TABLE requests. If you do not specify the key date, the current date is used as the key date. The contents of all time-dependent hierarchies in the cube will be exposed in the report as they were at the specified key date. To report against a specific concurrent version of the hierarchy, reference the field names associated with that version of the hierarchy in your request. This has to be coordinated with the key date that you specify. If you do not specify a correct key date (the one within the time period associated with the specified concurrent version of the hierarchy) your report will fail with an error message indicating absence of the specified hierarchy.

Example: Time Dependent Hierarchies in a Master File

The following portion of a Master File was produced using the key date 20081031 against the $0SD_C01 cube. Time-dependent hierarchies are defined for the 0MATERIAL dimension. The concurrent versions of the time dependent hierarchy are indicated in bold:

```
 DIMENSION=[0MATERIAL], CAPTION='Material', $
  HIERARCHY=[0MATERIAL], CAPTION='Material', HRY_DIMENSION=[0MATERIAL],
HRY_STRUCTURE=STANDARD, $
  HIERARCHY=[0MATERIAL                    001], CAPTION='Material
class', HRY_DIMENSION=[0MATERIAL], HRY_STRUCTURE=RECURSIVE, $
  HIERARCHY=[0MATERIAL                    MAT_CLASS_TD
V1], CAPTION='Material Class Time Dep', HRY_DIMENSION=[0MATERIAL],
HRY_STRUCTURE=RECURSIVE, $
  HIERARCHY=[0MATERIAL                    MAT_CLASS_TD
V2], CAPTION='Material Class Time Dep', HRY_DIMENSION=[0MATERIAL],
HRY_STRUCTURE=RECURSIVE, $
  HIERARCHY=[0MATERIAL                    PRDHA], CAPTION='Product
Hierarchy for material MARA', HRY_DIMENSION=[0MATERIAL],
HRY_STRUCTURE=RECURSIVE, $
  HIERARCHY=[0MATERIAL                    RCV], CAPTION='Retail category
view', HRY_DIMENSION=[0MATERIAL], HRY_STRUCTURE=RECURSIVE, $
```

The following portion of the Master File shows the fields listed for the two concurrent versions of the time dependent hierarchy. Note that the fields listed for the first version have the characters TIME_DEP in their names, and the fields listed for the second version have the characters TIME_DEP1 in their names:

Fields for Version 1 of the time dependent hierarchy:

```
FIELDNAME=MATERIAL_CLASS_TIME_DEP, USAGE=A143, ACTUAL=A143,
  MISSING=ON,
  TITLE='Material Class Time Dep Member Unique Name',
  WITHIN='*[0MATERIAL          MAT_CLASS_TD          V1]',
  PROPERTY=UID,  $


FIELDNAME=MATERIAL_CLASS_TIME_DEP_NAME, USAGE=A64, ACTUAL=A64,
  MISSING=ON,
  TITLE='Material Class Time Dep Member Name',
  REFERENCE=MATERIAL_CLASS_TIME_DEP, PROPERTY=NAME,  $


FIELDNAME=MATERIAL_CLASS_TIME_DEP_LVLNO, USAGE=I2L, ACTUAL=I4,
  MISSING=ON,
  TITLE='Material Class Time Dep Member Level Number',
  REFERENCE=MATERIAL_CLASS_TIME_DEP, PROPERTY=LEVEL_NUMBER,  $


FIELDNAME=MATERIAL_CLASS_TIME_DEP_PARENT, USAGE=A143, ACTUAL=A143,
  MISSING=ON,
  TITLE='Material Class Time Dep Parent Unique Name',
  REFERENCE=MATERIAL_CLASS_TIME_DEP, PROPERTY=PARENT_OF,  $


FIELDNAME=MATERIAL_CLASS_TIME_DEP_PARENT_LVLNO, USAGE=I2L, ACTUAL=I4,
  MISSING=ON,
  TITLE='Material Class Time Dep Parent Level',
  REFERENCE=MATERIAL_CLASS_TIME_DEP, PROPERTY=PARENT_LEVEL_NUMBER, $
```

```
FIELDNAME=MATERIAL_CLASS_TIME_DEP_CHILDREN_CARD, USAGE=I9, ACTUAL=I4,
 MISSING=ON,
 TITLE='Material Class Time Dep Member Children Cardinality',
 REFERENCE=MATERIAL_CLASS_TIME_DEP, PROPERTY=CHILDREN_CARDINALITY, $
FIELDNAME=MATERIAL_CLASS_TIME_DEP_CAPTION, USAGE=A60, ACTUAL=A60,

  MISSING=ON,
  TITLE='Material Class Time Dep Member Caption',
  REFERENCE=MATERIAL_CLASS_TIME_DEP, PROPERTY=CAPTION,  $
```

Fields for Version 2 of the time dependent hierarchy:

```
FIELDNAME=MATERIAL_CLASS_TIME_DEP1, USAGE=A143, ACTUAL=A143,
  MISSING=ON,
  TITLE='Material Class Time Dep Member Unique Name',
  WITHIN='*[0MATERIAL             MAT_CLASS_TD            V2]',
  PROPERTY=UID,  $
FIELDNAME=MATERIAL_CLASS_TIME_DEP1_NAME, USAGE=A64, ACTUAL=A64,
  MISSING=ON,
  TITLE='Material Class Time Dep Member Name',
  REFERENCE=MATERIAL_CLASS_TIME_DEP1, PROPERTY=NAME,  $
FIELDNAME=MATERIAL_CLASS_TIME_DEP1_LVLNO, USAGE=I2L, ACTUAL=I4,
  MISSING=ON,
  TITLE='Material Class Time Dep Member Level Number',
  REFERENCE=MATERIAL_CLASS_TIME_DEP1, PROPERTY=LEVEL_NUMBER,  $
FIELDNAME=MATERIAL_CLASS_TIME_DEP1_PARENT, USAGE=A143, ACTUAL=A143,
  MISSING=ON,
  TITLE='Material Class Time Dep Parent Unique Name',
  REFERENCE=MATERIAL_CLASS_TIME_DEP1, PROPERTY=PARENT_OF,  $
FIELDNAME=MATERIAL_CLASS_TIME_DEP1_PARENT_LVLNO, USAGE=I2L,ACTUAL=I4,
 MISSING=ON,
 TITLE='Material Class Time Dep Parent Level',
 REFERENCE=MATERIAL_CLASS_TIME_DEP1, PROPERTY=PARENT_LEVEL_NUMBER, $
FIELDNAME=MATERIAL_CLASS_TIME_DEP1_CHILDREN_CARD, USAGE=I9,ACTUAL=I4,
 MISSING=ON,
 TITLE='Material Class Time Dep Member Children Cardinality',
 REFERENCE=MATERIAL_CLASS_TIME_DEP1, PROPERTY=CHILDREN_CARDINALITY,$
FIELDNAME=MATERIAL_CLASS_TIME_DEP1_CAPTION, USAGE=A60, ACTUAL=A60,
 MISSING=ON,
 TITLE='Material Class Time Dep Member Caption',
 REFERENCE=MATERIAL_CLASS_TIME_DEP1, PROPERTY=CAPTION,  $
```

Example: Reporting Against Time Dependent Hierarchies

The following request reports against a synonym named TIMEDEP that was created using the key date 20081031. It lists the net value of sales volume for the first version of the time dependent 0MATERIAL hierarchy. The request displays three levels of the hierarchy and only non-empty cells:

```
ENGINE BWBAPI SET KEY_DATE 20081031
TABLE FILE TIMEDEP
SUM NET_VALUE_OF_SALES_VOLUME
BY MATERIAL_CLASS_TIME_DEP_CAPTION HIERARCHY
SHOW TO DOWN 3
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

Material Class Time Dep Member Caption	Net Value of Sales Volume
ACCESSORIES	31,926,127.92
Parts	31,926,127.92
Deluxe Headlight	1,770,180.46
Deluxe Taillight	566,089.48
SunFun / 1200 cm3	29,589,857.98

The following request uses the second version of the time dependent hierarchy:

```
ENGINE BWBAPI SET KEY_DATE 20081031
TABLE FILE TIMEDEP
SUM NET_VALUE_OF_SALES_VOLUME
BY MATERIAL_CLASS_TIME_DEP1_CAPTION HIERARCHY
SHOW TO DOWN 3
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

Material Class Time Dep Member Caption	Net Value of Sales Volume
ACCESSORIES	1,770,180.46
Parts	1,770,180.46
Deluxe Headlight	1,770,180.46

Following is the same request against the version of the hierarchy that is not time dependent. The results will be the same for any key date when using this version of the hierarchy:

```
TABLE FILE TIMEDEP
SUM NET_VALUE_OF_SALES_VOLUME
BY MATERIAL_CLASS_CAPTION HIERARCHY
SHOW TO DOWN 3
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
GRID=OFF, $
END
```

The output is:

Material class Member Caption	Net Value of Sales Volume
Material class	145,209,499.79
Products	145,209,499.79
Computer systems	85,842,293.86
Computer hardware	17,874,768.15
Computer accessories	67,967,525.71
Paints / aux. and operating supplies	.00
Paints	.00
Pumps	59,367,205.93
Pumps (complete)	59,367,205.93

Reversing the Sign When Displaying a Measure

In a BEX query, one of the options available for a measure is to display its values with reversed plus and minus signs. If you know that the BEX query has specified that a measure should be displayed with reversed signs, you can use the DMC to create a synonym that enables display of that measure on WebFOCUS reports with reversed signs. This is done by adding the REVERSE_SIGN=ON attribute to the field declaration for that measure.

The signs are reversed whenever this attribute is present, whether the BEX query specified a reverse sign or not, so you must know the data and the query definition before applying this attribute. This attribute has no effect on the stored values of the measure, just their display.

Syntax: How to Display a Measure With Reversed Signs

Add the following attribute to the FIELD declaration for the measure to be displayed with reverse signs:

```
REVERSE_SIGN=ON
```

Example: Displaying a Measure With Reversed Signs

In the following partial listing of the ZMINUS synonym, the measure CM_1 has the REVERSE_SIGN=ON attribute. This attribute has been added because the BEX query specifies reversed signs for measure CM_1:

```
FILENAME=ZMINUS, SUFFIX=BWBAPI  , $
  SEGMENT=ZMINUS, SEGTYPE=S0, $
$ MEASURES FOR CUBE 0SD_C01/ZMINUS
  FIELDNAME=CM_1, ALIAS=4KN324NNXDL1055YDEO17L0L8, USAGE=D18.2,
    ACTUAL=D8, MISSING=ON,  TITLE='CM 1',
      REFERENCE=SUM, PROPERTY=MEASURE, REVERSE_SIGN=ON, $
  FIELDNAME=COST_OF_SALES____, ALIAS=4KN324VCGC6QIRPEJ8QDHMZB0,
    USAGE=D18.5, ACTUAL=D8, MISSING=ON,  TITLE='Cost of Sales (%)',
      REFERENCE=UNKNOWN, PROPERTY=MEASURE,  $
 DIMENSION=[0MATERIAL], CAPTION='Material', $
  HIERARCHY=[0MATERIAL], CAPTION='Material', HRY_DIMENSION=[0MATERIAL],
    HRY_STRUCTURE=STANDARD, $
  HIERARCHY=[0MATERIAL                  001], CAPTION='Material class',
    HRY_DIMENSION=[0MATERIAL], HRY_STRUCTURE=RECURSIVE, $
```

The following request against the ZMINUS synonym prints the measures CM_1 and Cost_of_Sales:

```
TABLE FILE ZMINUS
PRINT CM_1 COST_OF_SALES
BY MATERIAL_NAME
END
```

The partial output follows. The measure CM_1 is displayed with reverse signs:

PAGE 1		
Material Member Name	CM 1	Cost of Sales (%)
100-400	.00	.
1400-100	-486,395.92	72.52281
1400-200	-177,820.37	68.58794
1400-300	-15,185,915.01	48.67865
1400-310	491,315.31	103.81913
1400-400	-3,389,899.27	15.21928
1400-750	1,159,102.91	264.66759
AM2-GT	-76,574.25	3.77397
AZ2-600	2,550.45	104.82289
AZ2-730	-3,376.03	33.86726
BP-100	.00	.
C-100	.00	.
C-1100	.00	.
C-202	.00	.

Removing the REVERSE_SIGN=ON attribute from the synonym produces the following output for the same request:

PAGE 1		
Material Member Name	CM 1	Cost of Sales (%)
100-400	.00	.
1400-100	486,395.92	72.52281
1400-200	177,820.37	68.58794
1400-300	15,185,915.01	48.67865
1400-310	-491,315.31	103.81913
1400-400	3,389,899.27	15.21928
1400-750	-1,159,102.91	264.66759
AM2-GT	76,574.25	3.77397
AZ2-600	-2,550.45	104.82289
AZ2-730	3,376.03	33.86726
BP-100	.00	.
C-100	.00	.
C-1100	.00	.
C-202	.00	.

Reporting Rules

The reporting rules and features vary depending on the types of hierarchies defined in the synonym. These factors are summarized in the following sections.

Reference: Display Command and Prefix Operator Support

When BY HIERARCHY is not used in the request, the display command can be PRINT, even for aggregated data. It is internally converted to the appropriate command.

Note: To emphasize that the summary values are already in the cube and do not necessarily require any additional aggregation, the examples use the WRITE command, which is a synonym for SUM.

The measure operation is also automatically supplied internally. You do not have to specify a prefix operator in the request. However, if you do specify a prefix operator, it should match the measure aggregator.

For example, assume there is a measure called ASALES and another measure called SALES, where SALES is a simple sum and ASALES is an average. A request referencing SALES and ASALES can use the following display command:

```
WRITE SALES ASALES
```

The operation for the ASALES measure will automatically be converted internally to AVE.ASALES.

You can also specify the summarizing operation in the command:

```
WRITE SALES AVE.ASALES
```

In the synonym, the REFERENCE= attribute specifies the measure aggregator. For example, REFERENCE=SUM.

If you do specify prefix operators in the request, you can prevent the adapter from checking whether your operation matches the measure aggregator by issuing the following command:

```
ENGINE BWBABI SET CHECKAGGR OFF
```

Reference: Reporting Rules for All Hierarchies

Variables. SAP BW expects a query to contain values for variables. Values for all mandatory variables must be supplied. The method for specifying variable values is to include a WHERE or IF phrase in the TABLE request.

For example, the following WHERE phrase supplies the value 430 for the COMPANY_CODE variable:

```
WHERE COMPANY_CODE_N EQ '430'
```

BY. Lexicographic sort is available on any of the member properties as well as expressions (DEFINEs) based on properties, but the request must include a sort (possibly with the NOPRINT option) on a property that is guaranteed to uniquely identify a hierarchy member, either PROPERTY=UID or PROPERTY=NAME. BY HIERARCHY automatically adds a BY one of these properties. In a report referencing a level hierarchy, the request must specify one.

Hierarchies. Only one hierarchy from the same dimension can be represented on the report.

Cardinality. Cardinality is the number of members in a dimension, hierarchy, or hierarchy level. Because SAP BW is a multi-dimensional database, the volume of the report may grow as a product of cardinalities of the referenced dimensions unless you include a WHERE test to restrict the number of returned dimension members

Measures. Filtering on totals is done using WHERE TOTAL tests. Calculations on totals are done using COMPUTE.

Screening. Screening on member properties is available with and without sorting.

Reference: Reporting Rules for Parent/Child Hierarchies

BY HIERARCHY. Requires the command SUM, WRITE, or ADD. PRINT is not supported.

Hierarchical Context. The set of members selected for the report using a WHEN test is augmented so that every member of the hierarchy is shown within its hierarchical context. The displayed subtree of hierarchy members is a contiguous tree whose root is the root of the whole hierarchy. The added context members are shown without totals.

Report Compression. By default, hierarchy members with no fact data do not participate in the report. Hierarchy members for which no data passes screening do not participate in the report. In a multi-hierarchy report, the inner hierarchies are not shown for those members of outer hierarchies that appear on the report output only to show the context of selected members.

Full and Visual Totals. Visual totals for each node are composed from the totals of its hierarchical children shown on the same report. These are compatible with the totals shown for the same members in a multi-verb level-wise report. Visual totals are the default. The report can specify full totals by specifying the prefix operator FROLL on a measure field. Full totals are accumulated for each displayed hierarchy member individually. These are compatible with the totals shown for the same members in a non-hierarchical report on a parent-child hierarchy.

General Tips for Reporting

When creating a report request:

- PRINT and SUM can be used interchangeably when using level hierarchies.

  When reporting from Operational Data Store (ODS) or InfoSet Queries (ODS joins), depending on the summarization setting of the key fields, you may use the PRINT verb. When using PRINT against ODS objects, the BY fields (not to be confused with Key Figures Fields) determine the summarization level of unique values.

- To maximize reporting efficiency, always sort in the same direction as the hierarchy for the cube.

For example, if the hierarchy for a dimension is

```
Country > State_Province > City
```

sorting by City first, then State_Province results in an inefficient request. Also, because of the hierarchical structure of OLAP, you retrieve only one BY field for each SUM or PRINT field in the report.

SAP BW Support for Hierarchies

In SAP BW release 3.0A and earlier, one characteristic of the query can be expanded in a hierarchical display.

Reference: Creating Dynamic Hierarchies

The Adapter for SAP BW generates each level of a hierarchy with a field name corresponding to that level. For a dimension called Business Number and a hierarchy named States, level 1 will be States and level 2 will be the Business Number. You can create a report referencing all the fields of a hierarchy at once, or create it as an OLAP trigger report, referencing the top level of the hierarchy and enabling interactive drill down.

In SAP BW 3.0B and higher, multiple characteristics of the query can be expanded in a hierarchical display.

Creating Dynamic Dimensions

SAP BW, you can:

- Create dynamic dimensions by using the Create New Field (Define) function.
- Use hierarchy levels and dimensions that are not contained in the query cube, but are defined or calculated at run time.

Example: Creating a Dimension

The following request creates a dimension called Daily Sales, which concatenates the company name, the current date (picked up from an the &DATE variable), and the Sales figures for that date.

```
DEFINE Daily Sales/A124= company|&date|'Sales'
END

TABLE FILE MYQUERY
SUM Retail_Amount
BY Daily Sales
END
```

The output is:

```
Daily Sales                    Retail_Amount
ACME   11/15/2002                     10,000
SMCE   11/15/2002                     15,000
```

Managing SAP BW Metadata

SAP BW field names are generated from the Level Caption loaded in BW Master Data and obtained from the BW Query Cube. These captions provide friendly (person-readable) field names.

Note:

- **To report against an InfoProvider**, you must generate an SAP BEx Query and release it for OLAP *before* you can create a synonym. You will complete those tasks using the SAP Business Explorer, a tool that creates the SAP BW queries you will reference in your report requests. If you have not completed those tasks, see *Creating BEx Queries* on page 2148, then return to generating metadata.
- In order to create a synonym, you need permission on the function group SU_USER.
- **Accessing SAP BW Master Data tables.** Many organizations require analysis of Master Data in conjunction with Transactional Data in SAP BW:
  - Transactional Data changes frequently (for example, account data, sales and purchase orders, and inventory). This data is stored in InfoCubes and is directly accessible through the Adapter for SAP BW, where it is available for up-to-the minute analysis.
  - Master Data remains unchanged over a long period of time and is always needed in the same way (for example, access authorizations, standard printer information, and organization structures). However, this data is stored in SAP BW Master Data Tables that may not initially be accessible in the cube-based SAP BW environment.

  Since this complex analysis is predicated upon access to the Master Data, you must be able to make the Master Data tables accessible within the SAP BW environment. You can accomplish this by completing a simple configuration procedure using the table-based Adapter for SAP ECC.

Creating Synonyms

Synonyms define unique names (or aliases) for each SAP BW table or view that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Note: In order to create a synonym, you need permission on the function group SU_USER.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for SAP BW

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select Catalog

You can select a catalog from the drop-down menu. The default selection is *All Catalogs*.

Filter by Catalog and Cube

To display all available queries, leave the *Filter by Catalog and Cube* check box blank.

To filter by Catalog name, Cube name, or Description, click the check box. Three input boxes are displayed. Enter the name of the Catalog (InfoCube), Cube (Query Cube), or description in the appropriate box. You can use the wildcard character (*) as needed. For example:

```
0CCA*
```

When issuing a request against a query cube, the adapter creates the cube name from these components: the full catalog name, followed by a forward slash (/), followed by the query cube name.

If you enter a description, it must be a contiguous portion of the SAP BW description from the InfoCube or Query Cube.

Multilanguage

A synonym can be created using one or more languages. If you choose multiple languages, each selected language can have its own title and description attributes. For details on multilanguage titles and descriptions, see *Multilanguage Titles and Descriptions* on page 2218.

1. Click the *Multilanguage* check box to select languages in addition to the default logon language (the value of the SET LANGUAGE command at server startup). The list of available languages is displayed. (The default language is not on the list.)
2. You may select one or more additional languages.

   For SAP BW, you must be authorized for each language you wish to access. Your userid and password are stored in the sapserv.cfg. The languages are retrieved from your selections on this list.

Note, however, that for the your language selections to be implemented, the selected languages must be installed. Also NLS must be enabled and your language selections must be consistent with your code page settings in the NLS Configuration Wizard. To access this wizard from the Web Console menu bar, select *Workspace*, *Configuration*. In the navigation pane, expand the *General* folder and click *NLS*. For related information, see *Code Pages and Multilanguage Synonyms* on page 2219.

In addition, a Field Names option is displayed (see next item on chart for details).

Field Names

If you have selected Multilanguage synonyms, you can choose to use *Technical* or *Language*-based field names:

- *Technical* names are based on the global ID of the SAP BW object.
- *Language* names use the caption (description) of the BW object. Language-based name is the default setting.

If you choose Language-based, a Field Name case option becomes available (see next item on chart for details).

Field Name Case

For Language-based field names, choose Uppercase or Mixed-case:

- Uppercase displays all field names in the Master File component of the synonym in uppercase characters.
- Mixed-case displays field names in the Master File in the same case that is used by SAP BW. Mixed-case is the default setting.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Hierarchy Type

Select the type of hierarchy you want to create in the synonyms by choosing one of the following options from the *Hierarchy type* drop-down list:

- **Optimized.** This option creates synonyms in which hierarchies are represented as parent/child relationships except for the flat hierarchy. The flat hierarchy (the list of all leaf nodes whose parent is the root node) is represented as a level hierarchy with two fields, one with a field name of the form *hierarchy_name*_LEVEL_01 (containing the member's caption) and one with a field name of the form *hierarchy_name*_NAME (containing the member's name).

- **Level.** This option creates separate fields for each hierarchy level as in prior releases.

  Note: Existing synonyms from prior releases of the adapter may need to be migrated before you can use them in a request. To migrate a synonym, click its name on the metadata page and then click the *Migrate* option on the context menu.

Member Sampling

Check this box if you want the adapter to select all members of a hierarchy level to determine the optimum length for the field, and also the cardinality.

Note: Depending on the data volume, this may take a long time. If synonym creation takes too long, you may wish to turn the member sampling off.

When this box is unchecked, all field lengths for the hierarchy levels are set to alphanumeric 60, and the cardinality information will be missing.

Structured Dimensions

Structures consist of a group of characteristics and/or key figures. They can be saved and reused in other queries. You can represent SAP BW structures as dimensions or create a cartesian product of the fields in the structures. To include structures as dimensions in the synonym, check this box.

If you do not check this box, the created Master File includes fields representing the Cartesian product of all fields in the structures. All of these fields are placed under the MEASURES section of the Master File and treated as separate key figures.

Add Level0

Using Level Hierarchy Master Files, the top hierarchy node (the 0 level) in most cases contains just the *All member* or the *Not assigned* node. For reporting, these are rarely used in cases where the user is restricted by Hierarchy Node Authorization TYPE 1 and is using LEVEL HIERARCHIES in the metadata.

Key Date

Produces a Master File that contains all versions of the hierarchies (time dependent and not) that existed for the time period associated with the specified key date. Specifying the key date at the time of synonym creation makes sense mostly in the presence of a time-dependent hierarchy that has or has had multiple concurrent versions. The key date should be specified for a period of time when the hierarchy had the maximum number of concurrent versions.

If no key date is specified, the current date is used.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Select synonyms to generate

Click the radio button next to the name of each cube for which you want to create a synonym. To create synonyms for all cubes on the list, click the radio button above the list of cubes. The Default Synonym Name column specifies the synonym name that will be generated by default if you select that cube. To specify non-default names for the generated synonyms, overtype the default names.

To delete measures, dimensions, or hierarchies from the synonym, use the Data Management Console after the synonym has been created.

Reference: Multilanguage Titles and Descriptions

When you create a Multilanguage synonym, title and/or description attributes for each selected language are added to the synonym. For example, if you choose French, the synonym will contain additional TITLE and DESC attributes of the following form:

```
TITLE_FR='french title'
DESC_FR='french description'
```

The default logon language generates titles and descriptions using the TITLE and DESCRIPTION attributes.

- If you set the LANGUAGE parameter to a language for which you generated additional title and description attributes, titles and descriptions are generated with the TITLE_In and DESC_In attributes (In being the two character ISO code; for example, EN, DE, JA).

- If you set the LANGUAGE parameter to a language for which you did not generate additional titles or descriptions, the TITLE and DESCRIPTION attributes are used to generate titles and descriptions.

In order to make field names in the synonym language independent, they are created based on the field's unique name (these names are called technical names). In synonyms created using only the default language, field names are created based on the field caption.

Example: Code Pages and Multilanguage Synonyms

The languages you wish to select for multilanguage synonyms determine which code page you should configure. For example, ISO 8859-1 can accommodate most Western European languages. Therefore, using this code page, you can request English, German, French, and Spanish.

The Unicode UTF-8 code page (65001) supports all languages and can therefore be used with any combination of languages. However, the UTF-8 character encoding scheme uses three bytes (in ASCII environments) or 4 bytes (in EDCDIC environments) to represent characters, increasing the storage needed for character data.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Mapping Metadata

OLAP (OLE DB standard) uses the following terms for row sets: Hierarchies, Levels, Measures, Members, and Dimension Properties. The corresponding terms are mapped in BW and the Adapter for SAP BW.

It is a good practice to regenerate the adapter metadata whenever data is loaded into the cube. Regeneration keeps data up-to-date and enables the adapter to accurately record the extent to which each dimension hierarchy expands.

Reference: Mapping BW InfoProviders to OLE/DB Cubes

Business Information Warehouse (BW) InfoProviders are similar to OLE/DB cubes. Both are the central metadata objects and containers for data used for reporting. InfoPRoviders contain two types of data: Key Figures and characteristics. For related information, see *Creating BEx Queries* on page 2148. For examples of these mappings, see *Mapping Illustrations* on page 2220.

OLAP	BW	FOCUS
Cube	Query	Master File.
Dimension	Characteristic	Internal use appears as comment in Master File (see above).
Hierarchy	Hierarchy	Dimension (An Adapter for SAP BW Dimension is a group of fields connected by means of the WITHIN keyword. The WITHIN keyword relates inner values to outer values for summary purposes.
Level	Level	Field (the WITHIN keyword points to the parent field).
Member	Characteristic Value	Field values.
Dimension Property (including special property Key)	Attribute	Field (only printable when the reference Dimension Hierarchy is used as a BY field).
No OLAP mapping	SAP BW variable	Field (specially mapped in a Master File) that takes optional or required input.
Measure	Key Figure (Measure)	Numeric fields upon which FOCUS verbs (PRINT/SUM) can perform operations.

Example: **Mapping Illustrations**

The following lines of syntax illustrate the makeup of the Adapter for SAP BW metadata. Each is listed with its corresponding definition:

```
FILENAME=WAREHOUS, SUFFIX=BWBAPI,$
```

This denotes the cube descriptor. The FILENAME=*value* is generated from the query cube name.

```
SEGNAME=WAREHOUS, SEGTYPE=S0,$
```

This is the segment name (used internally in FOCUS only). The SEGNAME=*value* is generated in the same way as FILENAME=*value*.

`$ DIMENSION -MEASURES-`

This is the beginning of the measures section of the Master File. The Fact Table is logically a dimension and is displayed accordingly in the Master File.

`$ CARDINALITY=`*`n`*

The cardinality of a dimension defines how many members can be retrieved for a report. If the Cardinality of a dimension is high, you should consider a WHERE test to restrict the returned values.

```
FIELD=STORE_INVOICE, ALIAS='STORE INVOICE', USAGE=D10.2, ACTUAL=D8.2,
MISSING=OFF,$
```

This describes the measure. The first section in the Master File defines the fields that comprise the Fact Table for your cube.

`$ DIMENSION:` *`dimension_name`*

This is the Dimension section comment. The Adapter for SAP BW synonym logic puts dimension names in comments before descriptions of the corresponding hierarchies and increases the readability of the Master File. All columns that follow a dimension comment are members of the dimension.

Note: Multiple hierarchies for the same dimension display as separate dimension entries.

```
FIELD= PRODUCT_FAMILY, ALIAS='PRODUCT FAMILY', WITHIN= *PRODUCT,
USAGE=A20, ACTUAL=A20, MISSING=OFF,$
```

This is the top level of a dimension hierarchy. Each subsequent level of the hierarchy will have a corresponding field name. Each level of the hierarchy has a following comment line with the level unique cardinality (count of members for the level).

Note: The WITHIN=*value* cannot exceed 66 characters. If the hierarchy unique name is longer than 65 characters (plus 1 character for *), Create Synonym generates a message. Although OLAP supports summary levels (the root level in the hierarchy), these are not displayed in the Master File.

```
FIELD=STORE_MANAGER, ALIAS='STORE MANAGER', USAGE=A20, ACTUAL=A20,
MISSING=ON,$
```

This describes the Dimension property. Any fields listed in a dimension section of the Master File that do not have an assigned hierarchy (for example, no WITHIN), represent dimension properties. In this example, the property stores the manager name for the store in the rollup.

Reference: Syntax Conventions for Master Files

The following conventions must be followed in a Master File that is used with SAP BW:

- **JOIN Restriction.** Joins are not supported in convention with BW syntax. However, you produce the same results using MATCH FILE.
- **Quotes are required.** Any field or hierarchy name containing blanks must be enclosed in single quotation marks.
- **Hierarchy levels.** Hierarchy levels do not have clearly defined data types. As a result, the levels are always represented as alpha.
- **Special Property - Key.** BW dimensions have a default hierarchy noted by the name of the dimension appended with level_01. These default hierarchy fields have a special property called a Key that contains the BW internal reference code for the field. A Key may be printed for only the default hierarchy and may be used only when it is referenced. A Key field may also be used in WHERE or DEFINE statements, providing the dimension field is referenced. In the synonym, the field name for this field has the suffix _NAME. For related information, see *Reporting Rules for All Hierarchies* on page 2210.

Example: Dimension: 0MATERIAL

The following example shows what can be expected when selecting the fields in a report. It is important to remember that two fields are returned for the same data value.

```
Field:0MATERIAL_LEVEL01
Field:0MATERIAL_NAME
```

as displayed in a report:

```
0MATERIAL_LEVEL_01    0MATERIAL_NAME     TOTAL_SALES
Fitdrink 2000(CAN)    R100032            $1000.00
```

Both fields represent the *same* characteristic value.

0MATERIAL_LEVEL_01 is the dimension level containing the captions for all members of the dimension.

0MATERIAL_NAME contains the SAP BW technical name for the members of the dimension.

Variable Types

SAP BW supports four types of variables:

- **Members.** Can be displayed in a separate TABLE request. When used in a TABLE query, a search criterion must be supplied if the variable type is mandatory (and there is no default value). The search criterion must render a set of variable values that correspond to the variable, which may be single value, interval, or complex (for complex there is no restriction on combinations). Each search predicate (an AND logical expression) can reference only one variable and no other cube elements.
- **Nodes.** Can be displayed in a separate TABLE request. Node type variables have a single value selection type.
- **Hierarchies.** Can be displayed in a separate TABLE request. Hierarchy type variables have a single value selection type.
- **Numeric.** Cannot be displayed.

For related information, see *Reporting With Variables* on page 2197.

Customization Settings

The following topics describe advanced set commands that allow you to customize your Adapter for SAP BW.

Syntax: How to Specify Empty Report Cell Settings

```
ENGINE BWBAPI SET EMPTY {ON|OFF}
```

where:

`ON`

Indicates that empty cells are included in a report and all data is shown whether it is associated with a dimension member or not.

`OFF`

Indicates that empty cells are excluded from a report (if measures are referenced) and dimension data is not shown if no fact data is associated with the dimension member (if measures are not referenced). OFF is the default value.

Tip: You can change this setting from the Web Console by clicking *Data Adapters* in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu.

Syntax: How to Specify Member Sampling Settings

During the synonym creation, member sampling is used to create Master File fields for dimension levels that precisely fit dimension member captions.

```
ENGINE BWBAPI SET MEMBER_SAMPLING {ON|OFF}
```

where:

ON

Indicates that the adapter will select all members of a hierarchy level to determine the optimum length for the field, and also the cardinality. The default value is ON.

Note: Depending on the data volume, this may take a long time. If synonym creation takes too long, you may wish to turn the MEMBER _SAMPLING OFF.

OFF

Indicates that all field lengths for the hierarchy levels are set to alphanumeric 60 and the cardinality information will be missing.

Tip: You can change this setting from the Web Console by clicking *Data Adapters* in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu.

Syntax: How to Specify Block Size

SAP recommends fetching large row and cell sets by portions in order to avoid crashes in the SAP server. The BLOCK_SIZE setting sets the size of a portion.

```
ENGINE BWBAPI SET BLOCK_SIZE n
```

where:

n

Sets the size of a portion. 0 is the default value, which means all members of the result set are returned in one portion.

The value of this setting should not be changed unless there is a timeout. Contact Information Builders high-level tech support for further information on how to use this option.

Tip: You can change this setting from the Web Console by clicking *Adapters*, right-clicking the configured adapter, and choosing *Change Settings*. The Change Settings pane opens.

Syntax: How to Specify Property Restrictions

The ability to query on attribute values in the MEMBERS row set was made available with SAP BW 3.0B. (For details, see the SAP OSS Note 609314.)

```
ENGINE BWBAPI SET PROPERTY_RESTRICTIONS {ON|OFF}
```

where:

`ON`

Indicates that any tests on a dimension attribute will be sent to SAP.

It is recommended that you set PROPERTY_RESTRICTIONS to ON if your SAP BW release and patch level support the specifying restrictions on user-defined properties as selection tables. (Contact SAP support for exact list of releases and patch levels.)

`OFF`

Indicates that the filtering will be done on the WebFOCUS server. OFF is the default value.

Tip: You can change this setting from the Web Console by clicking *Adapters*, right-clicking the configured adapter, and choosing *Change Settings*. The Change Settings pane opens.

Syntax: How to Specify Unassigned Member Settings

Sometimes dimension members are not explicitly arranged in a hierarchy. In SAP BW, assigned members are considered to be descendants of a root node named *All*, and unassigned members are considered to be children of a second root node named *All Not Assigned*. By default, Not Assigned members are not included in a report. You can include them using the INCLUDE_UNASSIGNED setting.

```
ENGINE BWBAPI SET INCLUDE_UNASSIGNED {OFF|ACTIVE|ALL}
```

where:

`OFF`

Does not include Not Assigned members belonging to an active hierarchy in a target report. OFF is the default value.

`ALL`

Includes all Not Assigned members belonging to an active hierarchy in a target report.

ACTIVE

Includes Not Assigned members belonging to an active hierarchy in which the lowest level is referenced. Individual unassigned members are listed at the lowest level of the hierarchy. This value is supported for level hierarchies only.

If the query does not include the lowest level, the summary for the Not Assigned nodes will not be shown unless there is a specific WHERE/IF statement at the lowest level.

Tip: You can change this setting from the Web Console by clicking *Adapters*, right-clicking the configured adapter, and choosing *Change Settings*. The Change Settings pane opens.

Syntax: How to Disable the Checking of Aggregation Operators

If you specify prefix operators in the request, the adapter, by default, checks that they are compatible with the measure aggregators in the BW data. If any are not, a message is returned and the request is not processed. You can prevent the adapter from checking whether your operation matches the measure aggregator by issuing the CHECKAGGR OFF command

```
ENGINE BWBAPI SET CHECKAGGR {ON|OFF}
```

where:

ON

Validates each prefix operator against the corresponding measure aggregator in BW. If a prefix operator does not match the corresponding measure aggregator, a message displays and the request is not processed. ON is the default value.

OFF

Does not validate prefix operators against measure aggregators.

Tip: You can change this setting from the Web Console by clicking *Adapters*, right-clicking the configured adapter, and choosing *Change Settings*. The Change Settings pane opens.

Syntax: How to Report Against Time-Dependent Hierarchies

When you report against a synonym with a time-dependent hierarchy, you use the ENGINE BWBAPI SET KEY_DATE command to set a key date for subsequent TABLE requests.

```
ENGINE BWBAPI SET KEY_DATE yyyymmdd
```

where:

yyyymmdd

Is the key date for the TABLE requests that follow. If you do not specify the key date, the current date is used as the key date. If you do not specify a correct key date (the one within the time period associated with the specified concurrent version of the hierarchy), your report will fail with an error message indicating absence of the specified hierarchy.

Syntax: How to Enable Support for Increased Number of Instances in a Request

SAP has introduced a new set of OLAP BAPIs that support requests with more than one million instances per axis. The SET USE_RSR_MDX command enables the Adapter for SAP BW to take advantage of these new BAPIs.

```
ENGINE BWBAPI SET USE_RSR_MDX {OFF|ON}
```

where:

OFF

Does not support requests with more than one million instances per axis. OFF is the default value.

ON

Supports requests with more than one million instances per axis.

Syntax: How to Build SAP VARIABLES Clauses Using Literal Strings

There are two different ways the adapter generates a SAP VARIABLES clause:

- Using literal strings.
- Using member unique names.

Using literal strings is perceived as more efficient because in this case the Adapter does not have to interact with the BW server in order to read the members of the hierarchy to obtain their unique names. However, generation of literal strings may or may not be possible depending on the types of tests against the variable fields and/or the type of the associated hierarchy and the variable selection type. Also, the success of an MDX request with literals depends on BW internal aspects which are not clearly documented by SAP.

The following setting controls whether the adapter will attempt to build a SAP VARIABLES clause using literals before resorting to the technique with member unique names:

```
ENGINE BWBAPI SET USE_SAP_VARIABLE_LITERALS {ON|OFF}
```

where:

ON

Instructs the adapter to attempt to build a SAP VARIABLES clause using literals.

OFF

Instructs the adapter to build a SAP VARIABLES clause using member unique names. OFF is the default value.

Support for BEx Structures

Structures consist of a group of characteristics and/or key figures. They can be saved and reused in other queries. For example, if plan/actual comparisons are being used extensively, it is possible to save a key figure group that consists of 10 key figures arranged for plan/actual comparison. To be able to use the same set of key figures in a different query, instead of selecting 10 key figures individually, adding the structure to the BEx query automatically brings those key figures into the query.

Note that SAP supports a maximum of two structures in a BEx query and only one of the structures can be of type Key Figure.

You can create a synonym with structured dimensions by checking the Structured Dimensions check box when creating the synonym.

Example: Representing Structures as Dimensions or as a Cartesian Product

You can represent structures as dimensions or create a cartesian product of the fields in the structures. Consider a BEx query that uses 2 structures:

- One structure is called *Curr. Year vs. Prev. year Cumulated* and it includes two time dimensions: Current Year and Prev.Year.
- The second structure is called *Orders-Sales-Returns* and it includes four key figures: Incoming Orders, Open Orders, Sales Volume, and Returns.

Without structured dimensions, the created Master File includes eight fields representing the cartesian product of the two structures. All of these fields are placed under the MEASURES section of the Master File and treated as separate key figures. The Master File looks something like this:

```
FILENAME=BW31C/Z2STRUCTURES_1KF_ROW_1COL, SUFFIX=BWBAPI  , $
  SEGMENT=Z2STRUCT, SEGTYPE=S0, $
$  -MEASURES-
    FIELDNAME=CURRENT_YEAR_CUMULATED_INCOMING_ORDERS,
      ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2,
      ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Incoming Orders', $
```

```
    FIELDNAME=CURRENT_YEAR_CUMULATED_OPEN_ORDERS,
      ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2,
      ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Open orders', $


    FIELDNAME=CURRENT_YEAR_CUMULATED_SALES_VOLUME,
      ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E, USAGE=D18.2,
      ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Sales Volume', $


    FIELDNAME=CURRENT_YEAR_CUMULATED_RETURNS,
      ALIAS=3466HXATG6RG1BG9GSTCN65IR3RV6E,
      USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Current year cumulated Returns', $


    FIELDNAME=PREV__YEAR_CUMULATED_INCOMING_ORDERS,
      ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E,
      USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Incoming Orders', $


    FIELDNAME=PREV__YEAR_CUMULATED_OPEN_ORDERS,
      ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E,
      USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Open orders', $


    FIELDNAME=PREV__YEAR_CUMULATED_SALES_VOLUME,
      ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E,
      USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Sales Volume', $
    FIELDNAME=PREV__YEAR_CUMULATED_RETURNS,
      ALIAS=36AWN99ETKI46HMTOGIVHHSOJ3RV6E,
      USAGE=D18.2, ACTUAL=D8, MISSING=ON,
      TITLE='Prev. year cumulated Returns', $
$  DIMENSION 0DISTR_CHAN (Distribution Channel):
$  HIERARCHY  (Distribution Channel):
...
...
```

The following is a sample WebFOCUS request against this Master File:

```
TABLE FILE Z2STRUCTURES_1KF_ROW_1COL
SUM CURRENT_YEAR_CUMULATED_INCOMING_ORDERS
    CURRENT_YEAR_CUMULATED_OPEN_ORDERS
    CURRENT_YEAR_CUMULATED_SALES_VOLUME
    CURRENT_YEAR_CUMULATED_RETURNS
    PREV__YEAR_CUMULATED_INCOMING_ORDERS
    PREV__YEAR_CUMULATED_OPEN_ORDERS
    PREV__YEAR_CUMULATED_SALES_VOLUME
    PREV__YEAR_CUMULATED_RETURNS
BY 0DISTR_CHAN_KEY
END
```

With structured dimensions, the created Master File includes four key figures placed under the MEASURES section of the Master File and the *Curr. Year vs. prev. year Cumulated* structure are treated as a dimension. The Master File looks something like this:

```
FILENAME=BW31C/Z2STRUCTURES_1KF_ROW_1COL_STON, SUFFIX=BWBAPI  , $
  SEGMENT=Z2STRUCT, SEGTYPE=S0, $
$  -MEASURES-
    FIELDNAME=INCOMING_ORDERS, ALIAS=[3RV6ENA572BRXFKFCO2M2ZUWS],
      USAGE=D18.2, ACTUAL=D8, MISSING=ON, TITLE='Incoming Orders', $
    FIELDNAME=OPEN_ORDERS, ALIAS=[3RV6ENHTQ0XHG23VII4YD1TMK],
      USAGE=D18.2, ACTUAL=D8, MISSING=ON, TITLE='Open orders', $
    FIELDNAME=SALES_VOLUME, ALIAS=[3RV6ENPI8ZJ6YONBOC7AN3SCC],
      USAGE=D18.2, ACTUAL=D8, MISSING=ON, TITLE='Sales Volume', $
    FIELDNAME=RETURNS, ALIAS=[3RV6ENX6RY4WHB6RU69MX5R24], USAGE=D18.2,
      ACTUAL=D8, MISSING=ON, TITLE='Returns', $
$  DIMENSION 0DISTR_CHAN (Distribution Channel):
$  HIERARCHY  (Distribution Channel):
...
...
$  DIMENSION EAZ5PAJIWG37DI674JZI2C3UB (Curr. year vs. prev. year
(cumulated to last period)):
$  HIERARCHY  (Curr. year vs. prev. year (cumulated to last period)):
    FIELDNAME=CURR__YEAR_VS__PREV__YEAR__CUMULATED_TO_LAST_PERIOD_,
      USAGE=A22, ACTUAL=A22, MISSING=ON,
      TITLE='Curr. year vs. prev. year (cumulated to last period)',
      WITHIN='*[EAZ5PAJIWG37DI674JZI2C3UB]', $
$  Cardinality: 2
$  DIMENSION EAZ5PAJIWG37DI674JZI2C3UB, PROPERTIES:
    FIELDNAME=EAZ5PAJIWG37DI674JZI2C3UB_KEY, ALIAS=MEMBER_NAME,
      USAGE=A64, ACTUAL=A64, MISSING=ON,
      TITLE='Curr. year vs. prev. year (cumulated to last period) Key', $
```

The following is sample WebFOCUS request against this Master File:

```
TABLE FILE Z2STRUCTURES_1KF_ROW_1COL_STON
PRINT INCOMING_ORDERS
    OPEN_ORDERS
    SALES_VOLUME
    RETURNS
BY CURR__YEAR_VS__PREV__YEAR__CUMULATED_TO_LAST_PERIOD_
END
```

Important: If the BEx query includes only two structures and no dimensions as free characteristics or no dimensions in rows and columns, you should use structured dimensions. Without structured dimensions, the created synonym will have only measures and no dimensions. Any WebFOCUS requests created against such a synonym would produce the following messages:

```
(FOC11205) ERROR IN MASTER FILE DESCRIPTION:

(FOC11229) MDX SEGMENT DESCRIPTION CONTAINS NO HIERARCHIES
```

Producing SAP BW Requests Using SQL

The following requests and output are examples of the capabilities of the Adapter for SAP BW using standard ANSI SQL code.

These examples require that you have created a synonym on the relevant tables from the Web Console. You can use the Web Console to navigate the SAP application hierarchy by searching the relevant tables or by doing a simple search by name.

Note: SQL does not support hierarchical reporting (BY HIERARCHY).

Example: Retrieving Values From SAP BW

The following syntax retrieves values from an SAP BW database:

```
SELECT SUM(DISBURSEMENT),ADDRESS1_LEVEL_01 FROM ZPAY4
GROUP BY ADDRESS1_LEVEL_01
ORDER BY ADDRESS1_LEVEL_01;
```

The output is:

address1 Level 01	disbursement
1106 BROADWAY	132,242.33
11607 PACIFIC BLVD	9,174.96
11607 TREMONT AVENUE	14,849.56
1200 RIVERDALE	9,983.99
187 WEST 39TH ST	9,209.97
2601 8TH AVENUE	26,274.65
31-18 SOUTHERN BLVD	1,165.64
719 SEVENTH STREET	10,800.00
DRAWER 4, COOPER STA	21,406.32
P.O. BOX 1206	53,636.74
P.O. BOX 17616	6,703.90

Example: Selecting Items From a Hierarchy

The following syntax selects items from a hierarchy in MARA:

```
SELECT SUM(SALES_VOLUME),SUM(INCOMING_ORDERS),
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02 FROM ZBIGQ
GROUP BY PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02
ORDER BY PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02
```

The output is:

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Sales Volume	Incoming Orders
Foods	Bakery product; configured to order	.00	400.00
Hardware	PCs	85,159,860.98	86,754,997.71
	Printer	.00	60,011.10
Lighting	Bulbs	62,235,898.00	62,674,320.00
Machines	Pumps	44,328,485.55	46,231,307.55
Paints	Gloss paints	.00	125.03
Services	Maintenance	291,562.54	485,315.83
Vehicles	Cars	155,640.00	155,640.00
	Motorcycles	41,761,619.76	42,494,811.00

Example: Selecting a Complete Hierarchy Path

The following syntax retrieves a complete hierarchy path from MARA:

```
SELECT SUM(SALES_VOLUME),SUM(INCOMING_ORDERS),
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04
FROM ZBIGQ
GROUP BY
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04
ORDER BY
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_01,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_02,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_03,
PRODUCT_HIERARCHY_FOR_MATERIAL_MARA_LEVEL_04;
```

The output is:

Product Hierarchy for material MARA Level 01	**Product Hierarchy for material MARA Level 02**	**Product Hierarchy for material MARA Level 03**	**Product Hierarchy for material MARA Level 04**	**Sales Volume**	**Incoming Orders**
Foods	Bakery product; configured to order			.00	400.00

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
Hardware	PCs	Drive	Harddisk 1080 MB / SCSI-2-Fast	4,234,580.64	4,094,612.64
			Harddisk 2113 MB / ATA-2	5,216,171.71	5,216,171.71
			Harddisk 2149 MB / SCSI-2-Fast	4,727,005.36	4,713,724.40
			Harddisk 4294 MB / SCSI-2-Fast	5,132,812.80	5,132,812.80
		Input device	Professional keyboard - MAXITEC Model	720,022.65	714,750.89
			Professional keyboard - NATURAL Model	813,765.24	788,838.81
			Professional keyboard - PROFITEC Model	576,079.36	576,079.36
			Standard Keyboard - EURO Model	620,294.21	600,911.21
			Standard Keyboard - EURO-Special Model	633,094.19	631,433.36

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Memory	SIM-Module 16M x 32, 70 ns	808,663.10	808,663.10
			SIM-Module 4M x 36, 70 ns	348,434.93	337,138.13
			SIM-Module 8M x 32, PS/ 2-72 Pin EDO-RAM	562,619.24	561,208.39
			SIM-Module 8M x 36, 70 ns	373,794.70	373,794.70

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Monitor	Flatscreen LE 50 P	1,116,349.10	1,155,586.30
			Flatscreen LE 64P	1,749,563.60	1,749,563.60
			Flatscreen MS 1460 P	2,525,626.85	2,704,385.95
			Flatscreen MS 1575P	2,375,663.66	2,469,023.66
			Flatscreen MS 1585	3,365,928.24	3,487,309.32
			Flatscreen MS 1775P	3,809,366.74	4,074,466.82
			Flatscreen MS 1785P	4,722,113.10	4,722,113.10
			Jotachi SN4000	4,396,902.60	4,572,792.60
			Jotachi SN4500	2,473,317.57	2,647,587.95
			Jotachi SN5000	2,949,233.80	2,949,233.80
			MAG DX 15F/Fe	2,437,207.85	2,521,555.85
			MAG DX 17F	2,567,506.69	2,655,601.89
			MAG PA/DX 175	2,976,662.21	3,181,471.75

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
			PAQ MONITOR, 17", Color	683,666.40	685,463.40
			PAQ Monitor, 20", Color	1,371,234.90	1,371,234.90
			SEC Multisync XV 17	3,557,594.77	3,690,080.79
			SEC Multisync XV15	2,961,072.00	2,961,072.00
			Sunny Extreme	3,215,185.42	3,339,353.42
			Sunny Tetral3	2,996,901.84	2,996,901.84
			Sunny Xa1	2,830,258.78	3,026,866.52
		PC ensemble	Maxitec-R 3100 Personal Computer	3,616,021.08	3,600,521.08
			Maxitec-R 375 personal computer	194,520.00	194,520.00
		Processor	Motherboard 3100	.00	18,558.62
			Processor 100 MHz	401,608.95	401,608.95
			Processor 133 MHz	15,949.50	11,574.50
			Processor 166 MHz	1,083,067.20	1,016,409.60

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
	Printer	Laser printer	High Speed Printer	.00	60,011.10

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
Lighting	Bulbs	Light Bulb 40 Watt clear 220/235V		15,351,358.00	15,351,358.00
		Light Bulb 40 Watt frosted 220/235V		5,432,920.00	5,256,060.00
		Light Bulb 40 Watt red 220/235V		2,976,965.00	3,197,425.00
		Light Bulb 40 Watt yellow 220/235V		3,032,064.00	2,917,008.00
		Light Bulb 60 Watt clear 220/235V		6,312,877.00	6,731,493.00
		Light Bulb 60 Watt frosted 220/235V		6,115,488.00	5,902,704.00
		Light Bulb 60 Watt red 220/235V		2,775,919.00	2,975,767.00
		Light Bulb 60 Watt yellow 220/235V		2,958,303.00	2,850,624.00
		Light Bulb 80 Watt clear 220/235V		7,095,740.00	7,589,438.00
		Light Bulb 80 Watt frosted 220/235V		5,120,425.00	4,935,200.00

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
		Light Bulb 80 Watt red 220/235V		2,416,380.00	2,416,380.00
		Light Bulb 80 Watt yellow 220/235V		2,647,459.00	2,550,863.00

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
Machines	Pumps	Special pump	Pump GG IDESNORM 100-200	5,798,946.55	6,191,646.55
			Pump cast steel IDESNORM 150-200	6,721,540.00	6,962,580.00
			Pump cast steel IDESNORM 170-230	9,558,837.00	9,565,269.00
			Pump chrome-steel IDESNORM 150-200	475,602.00	495,702.00
			Pump standard IDESNORM 100-402	6,893,740.00	7,336,070.00
			pump CR IDESNORM 150-200 ATO	7,822,280.00	8,343,320.00
			pump sphere-cast IDESNORM 150-200	7,057,540.00	7,336,720.00
Paints	Gloss paints	Opaque	Coating Matt Green RAL 6014/10 Liter	.00	125.03

Product Hierarchy for material MARA Level 01	Product Hierarchy for material MARA Level 02	Product Hierarchy for material MARA Level 03	Product Hierarchy for material MARA Level 04	Sales Volume	Incoming Orders
Services	Maintenance	HiTech maintenance	PC Service (Configurable)	171,992.54	285,995.83
			PC Service Plus	119,570.00	199,320.00
Vehicles	Cars	Car (complete)	SAPSOTA FUN DRIVE 2000GT	155,640.00	155,640.00
	Motorcycles	Accessories	Motorcycle Helmet - Standard	3,651,535.18	3,800,706.24
		Components	Deluxe Gas Tank Striping Decals	658,897.52	703,373.20
			Deluxe Headlight	1,609,718.67	1,675,515.07
			Deluxe Taillight	546,588.47	579,120.40
		Motor-cycle (compl.)	CrossFun / 350 cm3	10,842,211.75	11,243,347.19
			IDES Glad Boy configurable	929,800.00	953,830.00
			SunFun / 1200 cm3	23,522,868.17	23,538,918.90

Chapter 91

Using the Adapter for SAP ERP

The Adapter for SAP provides DataMigrator, WebFOCUS, and other enabled client applications access to SAP data sources. Client requests are dynamically converted into native SAP ABAP routines and return optimized answer sets to the requestor. For sample requests and output, see *Producing SAP Requests* on page 2299.

Note: The name SAP R/3 was used by SAP through Release 4.6. Other name variations in use by SAP are My SAP, SAP ECC, and SAP S/4 HANA. The WebFOCUS adapter services all of these variations.

In this chapter:

Preparing the SAP Environment

The Server requires the SAP RFC SDK to communicate with the SAP Application Server. The location of the RFC SDK depends on the platform specific search path.

Platform dependent search path for shared libraries (RFC SDK):

- **Windows:**

  Environment variable PATH.

- **AIX:**

  Environment variable LIBPATH.

- **z/OS:**

  Environment variable LIBPATH.

- **All other UNIX platforms:**

  Environment variable LD_LIBRARY_PATH.

  If the server does not start, and indicates that some NLS libraries cannot be found, try:

  ```
  EXPORT NLSUI_7BIT_FALLBACK=YES
  ```

You will use SAP GUI to logon to SAP and prepare the environment.

A Developer Key is required for the SAP user ID of the administrator. (The system prompts for the Developer Key if it has not been entered previously.)

USER IDs are required for authorization, with variations in privilege assignments based on the nature of the users responsibilities. Basic privileges are assigned to users who only need to execute procedures (focexecs). Administrative privileges are assigned to users who need to perform a variety of administrative functions for the Adapter for SAP. For example, the Adapter for SAP generates ABAP/4 programs dynamically at run time. This is done using adapter components uploaded into the SAP system during the configuration of the adapter, a task that requires administrative privileges.

For related information, refer to the following lists of processing objects: *List of Authorization Objects, Fields, and Required Values for USER* on page 2244 and *List of Authorization Objects for Background Processing* on page 2247.

Reference: List of Authorization Objects, Fields, and Required Values for USER

The following charts reflect any differences that apply for basic users and adapter administrators.

S_RFC: Authorization check for RFC access

Field	Value
ACTVT (Activity)	16 (Execute)
RFC_NAME (Name of RFC to be protected)	**For administrators:** AQRC, SDTX, SLST, SUTL, SYST, Z*** **For basic users:** SYST, Z***
RFC_TYPE (Type of RFC object to be protected)	FUGR (function group)

S_TCODE: Authorization check for Transaction Start

Field	Value
TCD (Transaction code)	**For administrators:** S001, SE09, SE11, SE11_OLD, SE13, SE37, SE38, SE80, SU53 **For basic users:** S001, SE37, SE38, SU53

S_ADMI_FCD: System Authorizations

Field	Value
S_ADMI_FCD (System administration functions)	**For administrators:** MEMO

S_C_FUNCT: C Calls in ABAP Programs

Field	Value
ACTVT (Activity)	**For administrators:** 16 (Execute)
CFUNCNAME (Name of a CALLable C routine)	**For administrators:** SYSTEM

Field	Value
PROGRAM (ABAP program name)	**For administrators:** SAPLSTRF, SAPLSTRI

S_DATASET: Authorization for File Access

Field	Value
ACTVT (Activity)	**For administrators:** 33, 34, A6
FILENAME (Physical file name)	**For administrators:** *
PROGRAM (ABAP program name)	**For administrators:** *

S_TABU_DIS: Table Maintenance (via standard tools such as SM30)

Field	Value
ACTVT (Activity)	**For administrators:** 03
DICBERCLS (Authorization group)	**For administrators:** SS, &NC&

S_DEVELOP: ABAP Workbench

Field	Value
ACTVT (Activity)	**For administrators:** 01, 02, 03, 07, 16, 40
DEVCLASS: Development class for transport system	**For administrators:** *
OBJNAME: Object name	**For administrators:** *
OBJTYPE: Object type	**For administrators:** DEVC, FUGR, PROG, TABL, TABT
P_GROUP: Authorization group with ABAP programs	**For administrators:** *

S_TRANSPORT: Transport Organizer

Field	Value
ACTVT (Activity)	**For administrators:** 01
TTYPE (Request type (Change and Transport System))	**For administrators:** DTRA, TASK

Reference: List of Authorization Objects for Background Processing

Field	Value
S_BTCH_ADM	N (unless you want the user to have administrative rights for Background jobs)
S_BTCH_JOB	RELE

Preparing the SAP Environment for Adapter Components

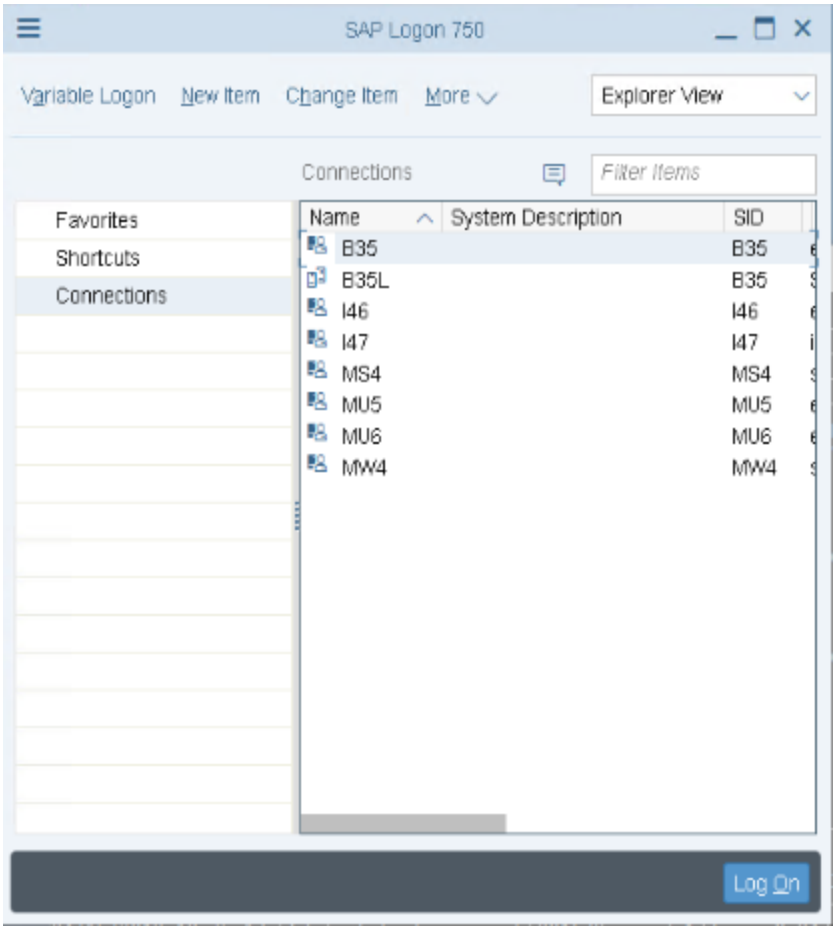
The Adapter for SAP generates ABAP/4 programs dynamically at run time. This is done using adapter components uploaded into the SAP system during the configuration of the adapter. Prior to uploading the components, the SAP system has to be prepared.

To prepare the SAP environment for the adapter components, perform the following steps:

1. Log on to SAP.
2. Create the Development Class or Package.
3. Create the Function Group.
4. Activate the Function Group.
5. Verify the Function Group.

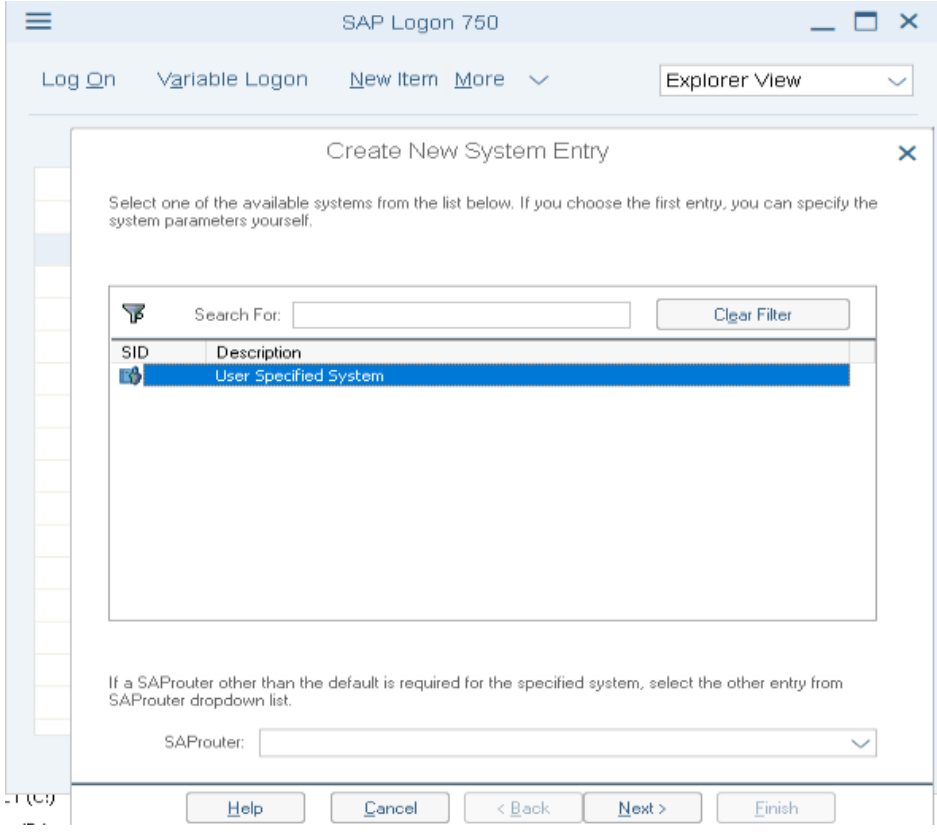
Procedure: How to Log on to SAP

1. Launch SAP logon. In a standard SAP GUI installation, the executable file resides in \Program Files\SAP\FrontEnd\SAPgui\saplogon.exe. The SAP Logon dialog box opens.



If the list is empty, create a new entry.

a. Click *New*. The New Entry dialog box opens.



b. Type the Description and Application Server, and select the System Number. In most cases, the SAP Router String is not needed, but check with your assigned SAP technical contact.

   In the Application Server field, type either the SAP application server network name or IP address.

   Click *OK* to create a new entry on the Logon list.

2. Select the entry on the list and click *Logon*.

   A logon page prompting for a user ID and password appears. Logon to the system. Note that if the user ID is being used for the first time, the system will ask for the password to be changed.

Procedure: How to Create a Development Class or Package

Note: If you are using Release 4.7, create a Package instead of a Development Class.

1. Execute transaction SE80 to start the Object Navigator.
2. Type the Development Class or Package name for the installation of the Adapter for SAP (for example, ZIBI).

   Note: The Development Class or Package name can be a maximum of 14 characters and must follow SAP naming conventions for customer objects.

3. Click *Display*  .

   The system responds: Development Class or Package ZIBI does not exist.

4. Click *Yes* to create the object.
5. Type a short description for the Development Class or Package, for example, *IBI-Created*.
6. Accept all displayed defaults and click *Create*.

   The system generates a unique Change and Transport System (CTS) request #.

7. Save this CTS request # for deployment of the adapter to other SAP systems.
8. Click *Continue*.

Procedure: How to Create a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. Type the same Development Class or Package name for the installation of the Adapter for SAP that you specified in *How to Create a Development Class or Package* on page 2250 (for example, ZIBI).
4. Click *Display*.

   The system responds: Function Group ZIBI does not exist.

5. Click *Yes* to create the object.
6. Enter a description for the Function Group.
7. Accept all displayed defaults and click *Save* (unless otherwise directed by an SAP Basis Administrator).
8. Under *Create Object Directory Entry*, type the same characters assigned to the Development Class and click *Save*.
9. Under *Prompt for transportable change request*, type the CTS request # assigned during the creation of the Development Class or Package.

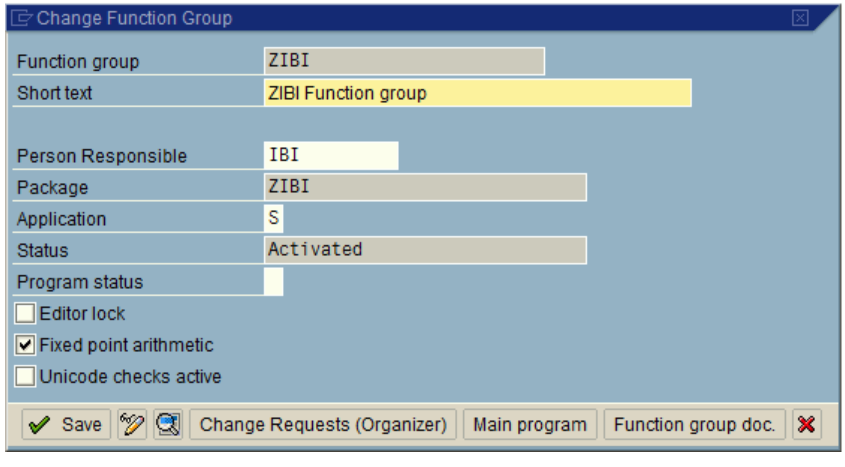
10. Click *Continue*.

Procedure: How to Deactivate the Unicode Checks

This procedure applies only to SAP Release 4.7.

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, type the name of the function group you created (for example, ZIBI).
4. Click *Display*.

   The object window shows an open folder with the name of the function group and a closed folder named Includes.
5. Right-click the function group name (for example, ZIBI) and select *Change* from the pop-up menu.



6. Deselect the *Unicode checks active* check box and click *Save*.

Procedure: How to Activate a Function Group

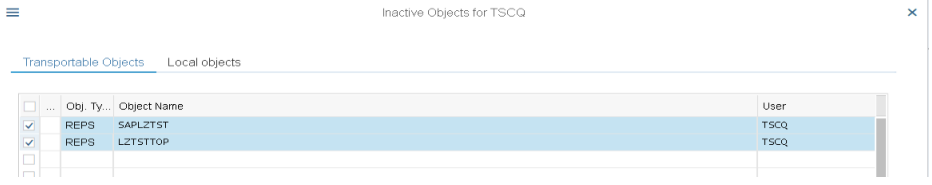
1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, type the name of the function group you created (for example, ZIBI).
4. Click *Display*.

The object window shows an open folder with the name of the function group and a closed folder named Includes.

5. Right-click the function group name (in this example, ZTST) and select *Activate* from the pop-up menu.

   A new window lists all inactive objects. Note that the object name now begins with SAPL.

   Make sure the object for the function group *ZTST* is selected.

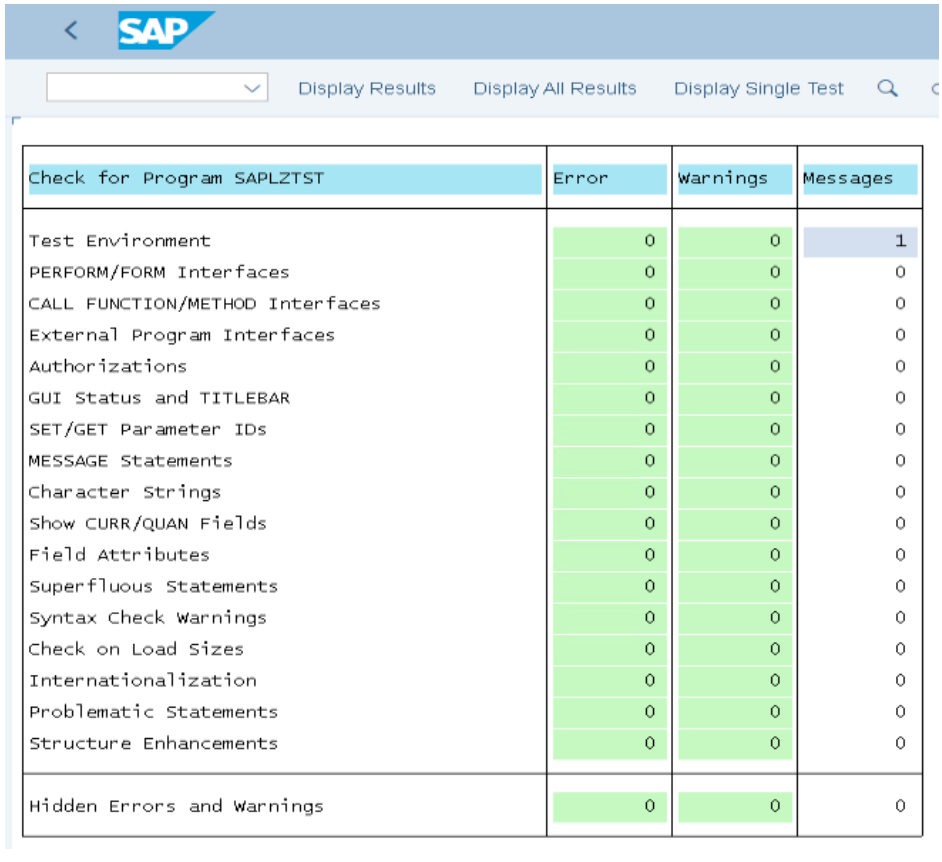


6. Click *Continue*.

Procedure: How to Verify a Function Group

1. Execute transaction SE80 to start the Object Navigator.
2. From the pull-down menu, select *Function Group*.
3. For Function Group value, type the name of the function group you created (for example, ZIBI).
4. Click *Display*.
5. Right-click the ZTST folder and select *Check - Extended Check*.
6. Select *Perform Check*.

You should get a report similar to the following:



Check for Program SAPLZTST	Error	Warnings	Messages
Test Environment	0	0	1
PERFORM/FORM Interfaces	0	0	0
CALL FUNCTION/METHOD Interfaces	0	0	0
External Program Interfaces	0	0	0
Authorizations	0	0	0
GUI Status and TITLEBAR	0	0	0
SET/GET Parameter IDs	0	0	0
MESSAGE Statements	0	0	0
Character Strings	0	0	0
Show CURR/QUAN Fields	0	0	0
Field Attributes	0	0	0
Superfluous Statements	0	0	0
Syntax Check Warnings	0	0	0
Check on Load Sizes	0	0	0
Internationalization	0	0	0
Problematic Statements	0	0	0
Structure Enhancements	0	0	0
Hidden Errors and Warnings	0	0	0

Note: If you encounter errors or warnings during this step, resolve them at the SAP layer before continuing with the installation of the Adapter for SAP.

7. Double-click the error line for detailed information about the error.

The SAP environment is now prepared for the next step of function module upload from the Web Console.

Reference: Customizing the Transport/Promotion of the ZXXX* Temporary Objects

During the installation of the SAP adapter, a set of function modules, including two temporary objects like ZXXXBTCH and ZXXXREPTS, are uploaded and registered in the SAP repository. In preparation for the transport from the QA to DEV to PROD application servers, the ZXXXBTCH object must be assigned to the ZXXX development class. Otherwise, ZXXXBTCH will be assigned to the development class $TMP or remain unassigned, either of which can cause the transport to fail.

Reference: Transport Control

SAP provides a transport control program (tp) to handle release upgrades and transports among SAP systems. The transport control program:

- Keeps track of transports.
- Exports and imports objects in the correct order.
- Ensures that imports into a target system are performed in the same order as the exports from the source system(s). (Processing of imports out of order can result in severe inconsistencies in the target system, which are difficult to diagnose.)
- Enables you to perform exports and imports separately.

During an export, the objects to be transported are extracted from the database of the source system and stored in files of the operating system.

During the import, the objects are added to the database of the target system (according to the transport function recorded in the task).

Note: For detailed technical background, refer to *http://help.sap.com*.

Accessing Multiple SAP Systems

The adapter can operate across multiple SAP systems. In SAP, data from multiple companies can share the same metadata, and is identified by a client value. Normally, a given system has multiple clients, and you may access one or more of them depending on your authorization. For each system, the adapter requires one SAP logon consisting of a client, a user, and a password. This logon must be:

- RFC-enabled.
- Authorized for all clients that you may access.

You may need additional authorizations depending on the SAP data you wish to access. Consult your SAP administrator for details.

Configuring the Adapter for SAP

Configuring the Adapter for SAP consists of specifying connection and authentication information.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

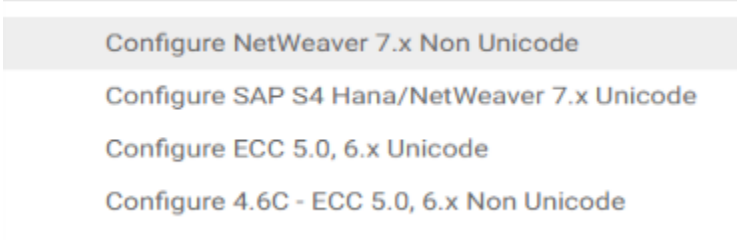
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for SAP

The *SAP* adapter is in the *ERP* group.

Right-click *SAP* on the Available Adapters page, and click a configuration option, such as SAP S4 Hana/NetWeaver 7.x Unicode, as shown in the following image.



The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

User Authentication Parameters

The major work of the adapter is to translate user requests into code that can be understood by SAP. For this purpose, an SAP user ID, with a given set of privileges, is required. In the following list, this user ID is referred to as IBI_USER.

System

Name of the connection; maximum 12 characters.

Package Parameters

FPOOL

Function group where the Adapter for SAP static function modules can be cataloged. This is the function group created in *Preparing the SAP Environment* on page 2243.

RANGEBEG

Reserved for Information Builders use.

RANGEEND

Reserved for Information Builders use.

Connection Parameters

The Load Balancing check box determines which of the following options is exposed.

GROUP

Name of the application group. An application group defines a list of application servers, on which an RFC application can be running. SAP transaction SMLG can be used to view or modify application groups.

Note that the entries are case-sensitive and blank spaces are significant.

Appears only when Load Balancing is checked.

MSGHOST

Host name of the SAP system (message server).

Appears only when Load Balancing is checked.

R3NAME

System ID of the SAP system.

Appears only when Load Balancing is checked.

HOST

Host name of the SAP application server.

GWHOST

Host name of the machine where the SAP gateway process is running. In the case where there is only one SAP application server, gwhost and host is the same.

SYSNR

SAP system number. This is a two-digit numeric value. Obtain this value from the SAP Administrator.

CONNECTION LANGUAGE

For SAP ECC Unicode configurations, you can select a language that is different from the language you use when logging on to the server (as determined by your NLS configuration).

From the CONNECTION LANGUAGE drop-down list on the Add SAP ECC non Unicode to Configuration pane, select the language you wish to use to connect to SAP ECC.

Note:

- When the CONNECTION LANGUAGE option is used to specify a language other than the one used for logging in to the server, the Create Synonym Multilanguage option is not available.

- For this option to work, the codepage of the iWay server must match the codepage of the SAP ECC server. For example, if the iWay server is configured FOR codepage 942 (SJIS), the SAP server must be configured for codepage 8000.
- There is no way to check if the requested connection language can be used by either the iWay server code page (either in terms of display or number of bytes) or the SAP ECC server.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

General User Login Parameters

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

CLIENT

SAP Client for the user logon, maximum 3 characters.

USER

SAP user ID for the user logon.

PASSWORD

SAP password for the user logon, maximum 8 characters.

Select the following check boxes, as required to:

- **Install Function Modules.** This option assumes that you have already completed the steps in *Preparing the SAP Environment* on page 2243, including creating a development class and Functions Group. To install the function modules for the adapter, select this check box. Ordinarily, function modules are installed only once. Reinstallation is required only if significant changes are made in the SAP system or if the Full-Function Server or the WebFOCUS Reporting Server is upgraded.
  - **Overwrite Existing (with the same fully qualified name).** This option is displayed when you select Install Function Modules. Use it to ensure that newly installed function modules replace any older versions with the same name.
- **Initialize Metadata.** Before you create a SAP synonym, you must provide base metadata in the form of SAP synonym candidates. The Initialize Metadata option creates the base metadata for you. Ordinarily, you only need to initialize metadata once. The only reason to reinitialize metadata is if you reinstall the function modules.

SNC Security Mode

When SNC is checked, you must specify values for the following additional parameters:

SNC_LIB

SNC_LIB contains the path to the external security product library. The external security product's library, external library, SNC_LIB, or gssapi library contains the functions provided by the external security product certified by SAP.

Set the environment variable SNC_LIB to contain the path to the security product library:

```
<drive>:\path\to\your\snclib.dll
```

SNC_ PARTNERNAME

Is the external name of the SAP system. This is an extended version of the external name called the SNC name. You create the SNC name by providing a prefix with the external user name that designates the name type entered as follows:

```
<SNC-name_of_SAP_AppServer>
```

For example:

```
p/secude:CN=miller,
OU=ADMIN, O=SAP, C=DE

p/krb5:miller@WDF.SAP-AG.DE
```

`SNC_QOP`

Indicates the level of protection.

`SNC_MYNAME`

Indicates the SNC name of the initiator, as in *own_snc_name*.

Note about SNC with load-balancing. Load-balancing or "group-logon" dynamically retrieves the target SNC-Name from the message server. When using SNC with load balancing, you must specify the following additional parameters:

```
SNC_PARTNERNAME=p:unused
```

```
SNC_LIB=<drive>:\path\to\your\snclib.dll
```

Parameters for Mixed Character Code Sets

The following section applies only when the server platform and the SAP instance platform do not use the same character code set (ASCII or EBCDIC).

Procedure: How to Configure the Adapter on IBM Systems Using EBCDIC Character Sets

To configure the adapter on IBM systems using the EBCDIC character set, the following code pages must be installed on the SAP server:

- In a z/OS environment, SAP code page 0126, or the correct code page for your language environment.
- For iSeries, SAP code page 0123, or the correct code page for your language environment.

1. Create two conversion tables as described in *Supporting Mixed Code Page Environments*, and transfer the tables to the $EDACONF/etc directory.

   For example, if the Server environment uses the code page 1100, then two conversion tables, 11000126.CDP and 01261100.CDP, should be created and transferred to the $EDACONF/etc directory.

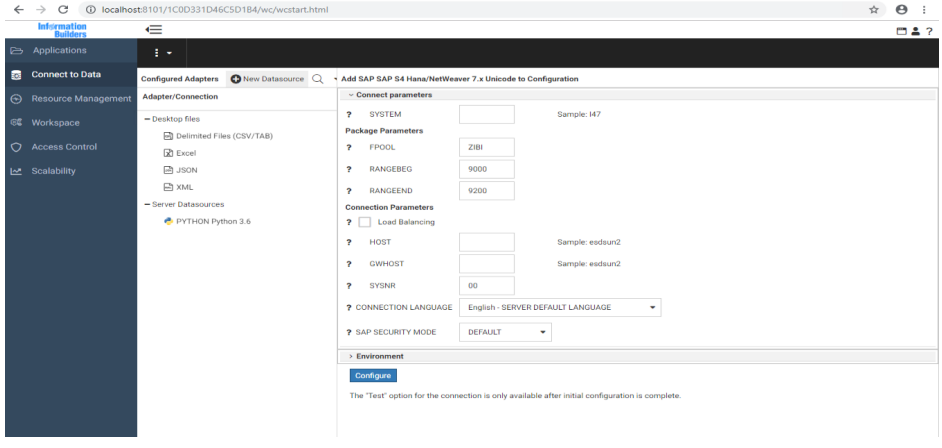
2. The following environment variables are required at run time, assuming your code page is 0126:

   ```
   export SAP_CODEPAGE=0126
   export PATH_TO_CODEPAGE=$EDACONF/etc/
   ```

   Note: Make sure to include the trailing "/" for the value of the PATH_TO_CODEPAGE parameter and export the variables SAP_CODEPAGE and PATH_TO_CODEPAGE.

Procedure: How to Configure the Adapter for SAP ECC or SAP S4 Hana

1. From the Web Console sidebar, click *Connect to Data*.
2. If the Adapter for SAP ECC or SAP S4 Hana has not yet been configured, click *New Datasource*. Select the *ERP* group, right-click *SAP*, click a version of SAP, then click *Configure*. The Add SAP to Configuration pane opens, as shown in the following image.



Tip: If the Adapter for SAP ECC or SAP S4 Hana has already been configured and appears in the list of Adapters in the navigation pane, you can customize the configuration by clicking the current system connection and choosing *Properties*. The Change Connection Parameters page opens, displaying the same parameters you would see on the Add to Configuration pane.

3. To establish the connection for the Adapter for SAP ECC or SAP S4 Hana enter values for the following configuration parameters.

`SYSTEM`

Enter a descriptive name of up to 12 characters to specify the connection to the SAP BW system.

`HOST`

Enter the host name of the server on the SAP BW system.

Important: This value, along with those for GWHOST and SYSNR, must *match* the corresponding values on the SAP GUI under System Entry Properties for SAP BW.

`GWHOST`

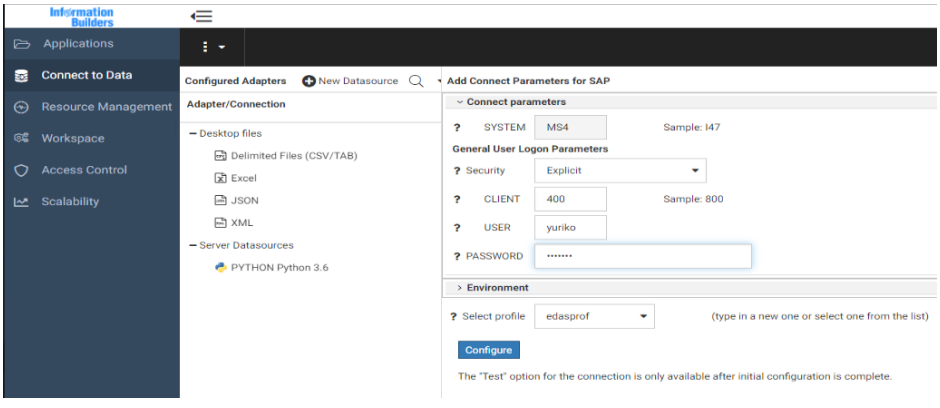
Enter the host name of the server where the SAP gateway process is running (in this case, the SAP BW system).

`SYSNR`

SAP system number. This is a two-digit numeric value.

4. Click *Configure*. You will see the message: *Connection successfully added to configuration*, and you will be prompted to configure a connection by clicking *Next*.

   The Add Connect Parameters for SAP page opens, as shown in the following image.



5. Enter or select the following connection attributes, and click *Configure*.

`Security`

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

`Client`

Is the SAP BW Client for the user logon.

`User`

Is the SAP BW authorization ID for the user logon.

`Password`

Is the SAP BW password for the user logon.

`Select profile`

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

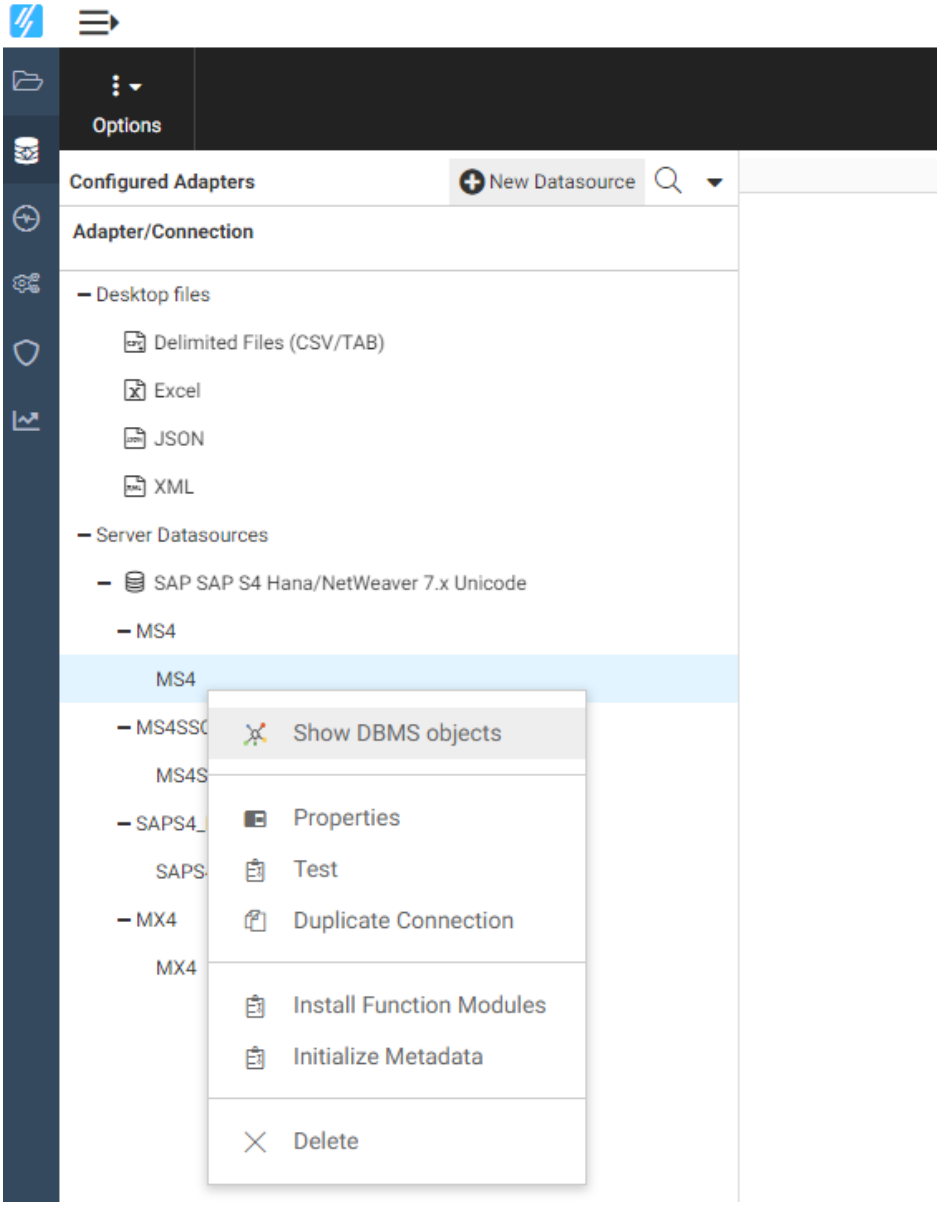
You will be prompted to install the function modules, if they are not installed.

6. Click *Next*.

   The configured adapter and connection are added to the Configured Adapters list.

   You can now click *Test* to verify the connection. If the connection between the SAP ECC or SAP S4 Hana adapter and the SAP BW environment has been established, sample data will be returned in the right pane. A positive result also verifies that the SAP BW Master Data tables you need will be accessible in the SAP BW environment.

You can now create a synonym for an SAP Table by right-clicking the connection and clicking *Show DBMS objects*, as shown in the following image.



The *Available Objects for SAP* page opens. You can select the TABLE objects for which to create synonyms.

Continue to use the Adapter for SAP, following the standard procedure for creating a synonym against a TABLE object. (For details, see *Creation Parameters for Table Synonyms* on page 2272.)

Controlling Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish. This setting enables you to control when the rfc api communications layer will be closed.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Control the Connection Scope

```
ENGINE SQLSAP SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLSAP`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Controlling the Transmission of COMMIT Requests to SAP ECC

The AUTOCOMMIT command controls when a COMMIT request is sent to SAP ECC.

Tip: You can change this setting manually or from the Web Console by clicking Data Adapters on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Control the Transmission of COMMIT Requests to SAP ECC

```
ENGINE SQLSAP SET AUTOCOMMIT ON {FIN|COMMAND}
```

where:

`SQLSAP`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Sends the COMMIT request to SAP only after the session has been terminated.

`COMMAND`

Sends a COMMIT request to SAP each time the command is issued. COMMAND is the default value.

Installing and Verifying SAP Components

After declaring connection attributes for SAP, to complete the configuration you must install and verify the SAP components.

Procedure: How to Install SAP Components

To install SAP components:

1. Launch the Web Console or the Data Management Console.
2. Access the list of configured adapters in the navigation pane.
3. Expand the *SAP* folder, then click the system you created when you set up user authentication parameters. For details, see *Connection Attributes for SAP* on page 2256.
4. Select *Properties* from the menu.

   The Change SAP System Connect Parameters pane opens. The Install SAP Components button is located at the bottom of the pane.
5. If you are re-installing, first select the *Overwrite existing Function Group* check box. A message warns that you are about to overwrite the function group, along with the associated function modules.

   If no error occurred, leave this box unchecked and proceed to step 6.

6. Click *Install SAP Components*. This process may take a few minutes.

   A list of the function modules that have been installed is displayed.

7. To confirm the installation, see *How to Verify a Function Group* on page 2252.

Procedure: How to Verify Installation of SAP Components

To confirm that all necessary function modules have been successfully installed, as described in *How to Install SAP Components* on page 2266, complete the following steps:

1. From SAP run transaction SE80.
2. Select *Function Group* from the first drop-down list.
3. Type the name of the Function Group you created when you declared connection attributes. For details, see *Package Parameters* in *Connection Attributes for SAP* on page 2256.
4. Click *Display* next to the text box.

   Folders named Function Modules and Includes should be under the Function Group name.

5. Expand the Function Modules and Includes folders to verify that the following appears:

   ```
   ZIBI_REP_CREATE
   ZIBI_REP_GET_B_D
   ZIBI_REP_ABORT_B
   ZIBI_REP_DELETE
   ZIBI_REP_RUN
   ZIBI_TEST_REVERSE
   ZIBI_DYNAMIC_RUN
   ZIBI_GET_APPL_TREE
   ZIBI_LDB_GET
   ZIBI_SE80
   ZIBI_REP_RUN_IN_B
   ZIBI_REP_GET_B_S
   ZIBI_DXOBJ_INFO
   ZIBI_DD_FKEYS_GET
   ```

   Note: Z*** represents the name of the Function Group. If the Development Class and Function Group name is ZIBI, then the Function Module name begins with ZIBI.

Post-Configuration Tasks in an SAP Environment

During the installation of the Adapter for SAP, a set of function modules in the development class (*ZXXX*) is added to the SAP repository.

For details, see *Customizing the Transport/Promotion of the ZXXXREPTS Temporary Object* on page 2268.

Customizing the Transport/Promotion of the ZXXXREPTS Temporary Object

SAP provides a transport control program (tp) for controlling release upgrades and transports among SAP systems. The transport control program :

- Keeps track of transports.
- Exports and imports objects in the correct order.
- Ensures that imports into a target system are performed in the same order as the exports from the source system(s). (Processing of imports out of order can result in severe inconsistencies in the target system, which are difficult to diagnose.)
- Enables you to perform exports and imports separately.

During an *export*, the objects to be transported are extracted from the database of the source system and stored in files of the operating system.

During the *import*, the objects are added to the database of the target system (according to the transport function recorded in the task).

Tip: For detailed technical background, navigate to *http://help.sap.com* (version 4.6C) and the menu path: *SAP Library, Basis Components, Change and Transport System (BC-CTS), Transport Tools (BC-CTS-TLS), Transport Control Program tp*. From here:

- Users should proceed to the section entitled *Preparing Operating System Users*.
- System administrators should proceed to the section entitled *Preparing the SAP System*.

Reference: Transport Data File Names

All data files are located in the transport directory data. The name of a data file consists of the name of the change request, and a code letter that distinguishes between data files generated by R3trans and those generated by application programs for application-specific development environment objects (ADOs):

- R <6 digits>.<source system> R3trans
- D <6 digits>.<source system> application programs

Reference: Change Request Information File

The change request information file contains information about a change request, including the transport type and the class of the objects to be transported. It also contains information about the steps required for the change request, exit codes, and the time of execution.

This information file is located in the cofiles directory. The name is derived from the name of the change request:

K <6 digits>.<source system>

Reference: **SAP Reserved Names**

The Adapter for SAP is compatible with the SAP Basis 4.6C version, the lowest certified level for the adapter. It reserves the following names:

- Development class Z*XXX*
- Function Group Z*XXX* and function module names with Z*XXX**

where:

XXX

Represents the 3 characters that are set by a certified SAP administrator to uniquely identify the client-independent table object and function modules. Note that a conflict with an existing unique name may cause the transport to fail.

Example: **Transporting a Request**

This illustration uses the sample request K900022 on P46. You may use it as a template for transporting a request.

Complete the following tasks in the order appropriate for each SAP system.

- Release the request from the sending system.
- Goto */usr/sap/transport/cofiles* and copy *K900022.P46* to the target system cofile directory.
- Goto */usr/sap/transport/data* and copy *R900022.P46* to the target system data directory.
- To check the transport system status of your SAP system, enter the command

  `tp checkimpdp <sapsid>`

  where:

  sapsid

  Is the system ID. For example, P46, which is defined during the SAP system installation process.

- Check the status of the tp import buffer with the command:

  `tp showbuffer <sapsid>`

Tip: Several tp commands are used in this procedure. For detailed information about tp procedures and commands, refer to the SAP tp documentation at *http://help.sap.com*.

- Clear the buffer using the command:

  ```
  tp cleanbuffer <sapsid>
  ```

- Check the status to make sure all items have been cleared using the command:

  ```
  tp showbuffer <sapsid>
  ```

- Add the transport using the command:

  ```
  tp add tobuffer P46K900022 <sapsid>
  ```

- Check that the transport has been added to the buffer using the command:

  ```
  tp showbuffer <sapsid>
  ```

- Import the transport using the command:

  ```
  tp import P46K900022 <sapsid> u1
  ```

  Note that you may need to add other options for override. For details, refer to the tp options documentation at *http://help.sap.com*.

- Verify that the Z*XXX* development class has been imported and activated, as indicated on your system screens.

Managing SAP Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server accesses, you create a synonym that describes the structure of the data source and maps the server data types to the SAP data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each SAP data object that is accessible from the server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File, which represent the server metadata.

Types of SAP Synonyms

The Adapter for SAP enables you to create synonyms for:

- **Tables**
- **Logical Data Bases (LDBs) and Native LDBs**

  For information that will help you determine when to use each type of LDB, see *Limitations for Logical Databases* on page 2278.

- **Business Application Programming Interfaces (BAPIs)**

  For information about requirements and limitations, see *Limitations for Function Modules and BAPIs* on page 2283.

- **Function Modules**

  For information about requirements and limitations, see *Limitations for Function Modules and BAPIs* on page 2283.

- **SAP Queries**

  An SAP query is a predefined report designed for users with little or no knowledge of the SAP programming language ABAP. From InfoSets (data sources), which are organized in user groups, an SAP query creates a list of information not already contained in the SAP system. This report may also include query variants within the global area. A variant is a saved set of selection criteria for a query or the report generated by the query. If you specify a variant when starting a query, the system uses the values in the variant for the query selection criteria.

Procedure: How to Create a Synonym

To create a synonym, you must have previously configured the adapter. You can create a synonym from the Web Console or the Data Management Console.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.

- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the synonym creation parameters reference.
   - *Creation Parameters for Table Synonyms* on page 2272.
   - *Creation Parameters for LDB Synonyms* on page 2276
   - *Creation Parameters for BAPI Synonyms* on page 2281.
   - *Creation Parameters for Function Module Synonyms* on page 2279.
   - *Creation Parameters for Query Synonyms* on page 2283.
4. After entering the parameter values, click *Create Synonym*.

   The Status pane indicates that the synonym was created successfully.

The synonym is created and added under the specified application directory.

Reference: Creation Parameters for Table Synonyms

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create the SAP synonym. These options appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Object Type

Select *TABLE* from the drop-down list

Multilanguage

A synonym can be created using one or more languages. If you choose multiple languages, each selected language can have its own title and description attributes. For details on multilanguage titles and descriptions, see *Multilanguage Titles and Descriptions* on page 2275.

1. Click the *Multilanguage* check box to select languages in addition to the default logon language (the value of the SET LANGUAGE command at server startup). You get a list of available languages (the default language is not on the list).
2. Select one or more additional languages. For SAP, the WebFOCUS Reporting Server logon language is used to retrieve all languages.

Note: For the your language selections to be implemented, the selected languages must be installed. Also NLS must be enabled and consistent with your code page settings in the NLS Configuration Wizard. To access this wizard from the Web Console menu bar, select *Workspace*, *Configuration*. In the navigation pane, expand the *General* folder and click *NLS*. For related information, see *Code Pages and Multilanguage Synonyms* on page 2276.

Find by Tables or Views

On the next pane:

- ❑ Select a Find by option: *Application Tree*, *Development Class*, or *Name*. (These options mirror the way SAP organizes and filters its data.)

  Subsequent selections vary depending on your *Find* method, as described below.
- ❑ Indicate whether to retrieve objects associated with Tables, Views, or both.

Application Tree. Choose this option, then click *Next* to retrieve a list of applications.

On the next pane, choose an application from the list, then click *Next*.

Development Class. Choose this option and type a Development Class name in the input box, or enter a string with wildcards, as needed. For example, enter: abc* (or abc%) to select development classes that begin with the letters abc; *abc (or %abc) to select those that end with the letters abc; *abc* (or %abc%) to select those that contain the letters abc at the beginning, middle, or end.

Click *Next* to display the Development Classe(s).

On the next pane, select a Development Class, then click *Next*.

Name. Choose this option and type the name of a Table in the input box, or enter a string with wildcards (* or %), as needed.

Enter a value in the Limit entry box. The default is 2000 tables. A message requesting a limit value is displayed when the list of available tables is too large to retrieve.

Click *Next*.

Choose the following options, as required:

Foreign keys

Click this check box if you wish to generate information about primary key and foreign key relationships in the Access File of the synonym.

An individual Master File is generated for each table. However, the relationships indicated in the Access File can be defined between any number of tables, and can reflect any number of levels of relationships among those tables.

Note: During synonym creation you may get one or more of the following messages:

```
(FOC44457) FOREIGN KEY FIELD **** CANNOT BE RESOLVED

(FOC44458) ONE OR MORE FIELDS OF THE FOREIGN KEY DO NOT EXIST

(FOC44459) MULTIPLE FOREIGN TABLES
```

where:

`****`

Is a table name.

These messages indicate that some of the foreign key relationships could not be resolved, and are, therefore, omitted during synonym creation. These messages are informational only: they should cause you no concern. In fact, the synonym is created and can be used in your requests without any problems.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

ODBC type fields

Select this check box only if the synonym is to be used by an application that connects to the WebFOCUS reporting server via ODBC.

Defines

When this option is selected, unsupported formats trigger the creation of DEFINE (virtual) fields.

From the list of tables at the bottom of the pane, choose those for which you wish to create synonyms:

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select objects for synonyms

Click the check box to the left of *Default Synonym Name* to generate synonyms for all listed tables, or check individual boxes in the same column to generate synonyms for the selected modules.

Reference: Multilanguage Titles and Descriptions

When you create a Multilanguage synonym, title and/or description attributes for each selected language are added to the synonym. For example, if you choose French, the synonym will contain additional TITLE and DESC attributes of the following form:

```
TITLE_FR='french title'
DESC_FR='french description'
```

The default logon language generates titles and descriptions using the TITLE and DESCRIPTION attributes.

- If you set the LANGUAGE parameter to a language for which you generated additional title and description attributes, titles and descriptions are generated with the TITLE_In and DESC_In attributes (In being the two character ISO code; for example, EN, DE, JA).
- If you set the LANGUAGE parameter to a language for which you did not generate additional titles or descriptions, the TITLE and DESCRIPTION attributes are used to generate titles and descriptions.

In order to make field names in the synonym language independent, they are created based on the unique name of the field (these names are called technical names). In synonyms created using only the default language, field names are created based on the field caption.

Example: Code Pages and Multilanguage Synonyms

The languages you wish to select for multilanguage synonyms determine which code page you should configure. For example, ISO 8859-1 can accommodate most Western European languages. Therefore, using this code page, you can request English, German, French, and Spanish.

The Unicode UTF-8 code page (65001) supports all languages and can therefore be used with any combination of languages. However, the UTF-8 character encoding scheme uses three bytes (in ASCII environments) or 4 bytes (in EDCDIC environments) to represent characters, increasing the storage needed for character data.

Reference: Creation Parameters for LDB Synonyms

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create the SAP synonym. These options appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Object Type

Select *LDB* from the drop-down list.

Multilanguage

A synonym can be created using one or more languages. If you choose multiple languages, each selected language can have its own title and description attributes. For details on multilanguage titles and descriptions, see *Multilanguage Titles and Descriptions* on page 2275.

1. Click the *Multilanguage* check box to select languages in addition to the default logon language (the value of the SET LANGUAGE command at server startup). You get a list of available languages (the default language is not on the list).
2. You may select one or more additional languages. For SAP, the WebFOCUS Reporting Server logon language is used to retrieve all languages.

Note: For the your language selections to be implemented, the selected languages must be installed. Also NLS must be enabled and your language selections must be consistent with your code page settings in the NLS Configuration Wizard. To access this wizard from the Web Console menu bar, select *Workspace*, *Configuration*. In the navigation pane, expand the *General* folder and click *NLS*. For related information, see *Code Pages and Multilanguage Synonyms* on page 2276.

Find by Native or Native Dynamic

On the next pane:

- Select a Find by option: *Application Tree*, *Development Class*, or *Name*. (These options mirror the way SAP organizes and filters its data.)

  Subsequent selections vary depending on your *Find* method, as described below.

- Indicate whether to retrieve objects associated with Native LDBs or Native Dynamic LDBs. For related information, see *Limitations for Logical Databases* on page 2278.

  In Native mode a static report is catalogued and executed.

  In Native Dynamic mode specific BAPIs are called to provide results on an LDB query.

Application Tree. Choose this option, then click *Next*.

On the next pane, choose an application from the list, then click *Next*.

Development Class. Choose this option and type a Development Class name in the input box, or enter a string with wildcards, as needed. For example, enter: abc* (or abc%) to select development classes that begin with the letters abc; *abc (or %abc) to select those that end with the letters abc; *abc* (or %abc%) to select those that contain the letters abc at the beginning, middle, or end.

Click *Next* to display the Development Classe(s).

On the next pane, select a Development Class, then click *Next*.

Name. Choose this option and type the name of a Table in the input box, or enter a string with wildcards (* or %), as needed.

Enter a value in the Limit entry box. The default is 2000 tables. A message requesting a limit value is displayed when the list of available tables is too large.

Choose the following options, as required:

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

From the list of LDBs at the bottom of the pane, choose those for which you wish to create synonyms:

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select objects for synonyms

Select the objects for which you wish to create synonyms:

- For all objects, click the check box to the left of the Default Synonym Names column.
- For specific objects, select the corresponding check boxes.

Reference: Limitations for Logical Databases

Synonym for SAP Logical Databases (LDB) can be generated in two different formats: SQL mode (set of joined SQL tables) or Native Mode (use of the underlying SAP Logical Database Program).

Note that in either mode, only the following LDBs are currently certified: VAV, SDF, KDF, CEK, CFK, CIK, CPK, CRK. Others need field certification.

- **SQL Mode.** Standard limitations concerning the number of Joins or number of fields in a synonym apply.

  From an end-user standpoint, SQL Mode usually runs faster, and is, therefore, the preferred method. Native Mode should be used only when there is a requirement to access so-called *structures*, for example, the segments SKC1A, SKC1C in SDF.

- **Native Mode.** Most LDBs in Native mode open a selection screen at run-time on which some of the parameters are required. Since the Adapter for SAP/R3-ECC works through RFC, it has no knowledge of this screen(s).

  This mode has two variations:

  - In Native mode, a static report is catalogued and executed.

- In Native Dynamic mode, specific BAPIs are called to provide results on an LDB query.

Reference: Creation Parameters for Function Module Synonyms

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for a Function Module. You click the *Create* button to generate the synonym based on your entries.

Note that not all function modules are suitable for reporting.

Object Type

Select *Function Module* from the drop-down list.

Multilanguage

A synonym can be created using one or more languages. If you choose multiple languages, each selected language can have its own title and description attributes. For details on multilanguage titles and descriptions, see *Multilanguage Titles and Descriptions* on page 2275.

1. Click the *Multilanguage* check box to select languages in addition to the default logon language (the value of the SET LANGUAGE command at server startup). You get a list of available languages (the default language is not on the list).
2. You may select one or more additional languages. For SAP, the WebFOCUS Reporting Server logon language is used to retrieve all languages.

Note: For the your language selections to be implemented, the selected languages must be installed. Also NLS must be enabled and your language selections must be consistent with your code page settings in the NLS Configuration Wizard. To access this wizard from the Web Console menu bar, select *Workspace*, *Configuration*. In the navigation pane, expand the *General* folder and click *NLS*. For related information, see *Code Pages and Multilanguage Synonyms* on page 2276.

Find by

On the next pane, select a Find by option: *Application Tree*, *Development Class*, or *Name*. (These options mirror the way SAP organizes and filters its data.)

Subsequent selections vary depending on your *Find* method, as described below.

Application Tree. Choose this option, then click *Next*.

On the next pane, choose an application from the list, then click *Next*.

Development Class. Choose this option and type a Development Class name in the input box, or enter a string with wildcards, as needed. For example, enter: abc* (or abc%) to select development classes that begin with the letters abc; *abc (or %abc) to select those that end with the letters abc; *abc* (or %abc%) to select those that contain the letters abc at the beginning, middle, or end.

Click *Next* to display the Development Class(es).

On the next pane, select a Development Class, then click *Next*.

Name. Choose this option and type the name of a Table in the input box, or enter a string with wildcards (* or %), as needed.

Enter a value in the Limit entry box. The default is 2000 tables. A message requesting a limit value is displayed when the list of available tables is too large.

Choose the following options, as required:

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

From the list of tables (modules) at the bottom of the pane, choose those for which you wish to create synonyms:

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select objects for synonyms

Click the check box to the left of *Default Synonym Name* to generate synonyms for all listed function modules, or check individual boxes in the same column to generate synonyms for the selected modules.

Reference: Creation Parameters for BAPI Synonyms

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create the SAP synonym. These options appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Object Type

Select *BAPI* from the drop-down list.

Multilanguage

A synonym can be created using one or more languages. If you choose multiple languages, each selected language can have its own title and description attributes. For details on multilanguage titles and descriptions, see *Multilanguage Titles and Descriptions* on page 2275.

1. Click the *Multilanguage* check box to select languages in addition to the default logon language (the value of the SET LANGUAGE command at server startup). You get a list of available languages (the default language is not on the list).
2. You may select one or more additional languages. For SAP, the WebFOCUS Reporting Server logon language is used to retrieve all languages.

Note: For the your language selections to be implemented, the selected languages must be installed. Also NLS must be enabled and your language selections must be consistent with your code page settings in the NLS Configuration Wizard. To access this wizard from the Web Console menu bar, select *Workspace*, *Configuration*. In the navigation pane, expand the *General* folder and click *NLS*. For related information, see *Code Pages and Multilanguage Synonyms* on page 2276.

Find by

On the next pane, select a Find by option: *Application Tree*, *Development Class*, or *Name*. (These options mirror the way SAP organizes and filters its data.)

Subsequent selections vary depending on your *Find* method, as described below.

Application Tree. Choose this option, then click *Next* to retrieve a list of applications that contain BAPIs.

On the next pane, choose an application from the list, then click *Next*.

Development Class. Choose this option and type a Development Class name in the input box, or enter a string with wildcards, as needed. For example, enter: abc* (or abc%) to select development classes that begin with the letters abc; *abc (or %abc) to select those that end with the letters abc; *abc* (or %abc%) to select those that contain the letters abc at the beginning, middle, or end.

On the next pane, select a Development Class, then click *Next*.

On the next pane, choose an object type, then click *Next*.

Name. Choose this option and type the name of a BABI in the input box, or enter a string with wildcards (* or %), as needed.

Enter a value in the Limit entry box. The default is 2000 tables. A message requesting a limit value is displayed when the list of available BAPIs is too large to retrieve.

On the next pane, select a BABI, then click *Next*.

Choose the following options, as required:

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

From the list of BABIs at the bottom of the pane, choose those for which you wish to create synonyms:

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select objects for synonyms

Select the objects for which you wish to create synonyms:

- For all objects, click the check box to the left of the Default Synonym Names column.
- For specific objects, select the corresponding check boxes.

Reference: Limitations for Function Modules and BAPIs

- A function module must have at least one Import or Export parameter.
- When filtering with an internal table of type Select Option, any combination of conditions can be applied.
- Filtering with an internal table that is not of type Select Option is not supported.

Reference: Creation Parameters for Query Synonyms

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a QUERY synonym. These options appear on multiple panes. To advance from pane to pane click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Object Type

Select *QUERY* from the drop-down list.

Cross Client Queries

SAP Queries can be created under two different query areas: Standard Area (client-specific) and Global Area (cross-client). By default, queries are created under the Standard Area so no action is required.

To create queries under the Global Area, select the *Cross Client Queries* check box.

Important: To be able to run WebFOCUS requests against queries that were originally created under the Global Area (cross-client), you must add the following selection criteria to the report request:

```
WHERE WORKSPACE EQ 'X';
```

Filter by Group and Query

To list queries for a specific Group, click the *Filter* checkbox and type a Group Name and/or a Query name in the input boxes provided, using the wildcard character (*) as needed.

Select Candidates button

Click to proceed to the next pane.

Note: If you receive the message No QUERYs Found, it is likely that the QUERY list was not generated.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Select objects for synonyms

Select the objects for which you wish to create synonyms:

- For all objects, click the check box to the left of the Default Table Names column.
- For specific objects, select the corresponding check boxes.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Example: SAP Query Synonym Variants

When creating an SAP query synonym, all variants (if any) will be found at CREATE SYNONYM time. A two-segment Master File is created. The top segment holds the VARIANT field for all possible variants. The second segment is the actual report fields. A variant must be specified for running the report.

For example, with the following synonym:

```
$     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$           SNAPpack generated on Thu Feb 20 11:44:07 2003
$           SAP instance: I46   SAP Release: 46C
$           MFD generated for QUERY: BASEAPP/ALM01_02
$     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
FILE=BASEAPP/ALM01_02  ,SUFFIX=SQLSAP,
  REMARKS='UG: 02 Query: ALM01',$
SEGNAME=VARIANT,SEGTYPE=S0,$
  FIELD=VARIANT, ALIAS=VARIANT,ACTUAL=A14,USAGE=A14,ACCEPT=
     'CAUS' OR
     'MY VARIANT' OR
     '            ',$


SEGNAME=REPORT,PARENT=VARIANT,SEGTYPE=S0,$
  FIELD=SKA1_SAKNR, ALIAS=SKA1_SAKNR, USAGE=A010, ACTUAL=A010,
  TITLE='G/L acct', DESC='G/L account number',$
  FIELD=SKA1_GVTYP, ALIAS=SKA1_GVTYP, USAGE=A002, ACTUAL=A002,
  TITLE='AT', DESC='P&L statement account type',$
  FIELD=SKC1C_BUKRS, ALIAS=SKC1C_BUKRS, USAGE=A004, ACTUAL=A004,
  TITLE='CoCd', DESC='Company Code',$
  FIELD=SKC1C_GJAHR, ALIAS=SKC1C_GJAHR, USAGE=A004, ACTUAL=A004,
  TITLE='Year', DESC='Fiscal year',$
  FIELD=SKC1C_GSBER, ALIAS=SKC1C_GSBER, USAGE=A004, ACTUAL=A004,
  TITLE='BA', DESC='Business Area',$
  FIELD=SKC1C_HWAER, ALIAS=SKC1C_HWAER, USAGE=A005, ACTUAL=A005,
  TITLE='Crcy', DESC='Currency Key',$
  FIELD=SKC1C_UM01O, ALIAS=SKC1C_UM01O, USAGE=P16.2, ACTUAL=A16,
  TITLE='SKC1C-UM01', DESC='Monthly balance',$
  FIELD=SKC1C_HWAER1, ALIAS=SKC1C_HWAER1, USAGE=A005, ACTUAL=A005,
  TITLE='Crcy', DESC='Currency Key',$
```

A valid request using variant 'CAUS' is:

```
TABLE FILE ALM01_02
PRINT
  SKA1_SAKNR
  SKA1_GVTYP
  SKC1C_BUKRS
  SKC1C_GJAHR
  SKC1C_GSBER
  SKC1C_HWAER
  SKC1C_UM01O
  SKC1C_HWAER1
IF VARIANT EQ 'CAUS'
IF READLIMIT EQ 50
END
```

Note that the ACCEPT values list generated in the Master File for SAP Query Synonym Variants is optional. While a variant must be specified for running a report, you can choose to maintain your own variant list.

Reference: Synonym Creation for IDOCs

In previous releases of the server (5.3 through 7.1), support for IDOC synonyms was integrated into the Web Console. In Version 7 Release 6.x, IDOC synonym creation functions have been removed from the Web Console. This includes EDA (Electronic Data Interchange)/ALE (Application Link Enabling) options, which are no longer available on the Web Console.

Support for the creation of IDOC synonyms is now limited to read-only. You can create synonyms for IDOCs using the method described below. However, while the adapter can read an external file that represents an IDOC, users are responsible for generating the external file and providing the DBC attributes associated with that external file. (For related information that will be useful when specifying DCB attributes for a flat file representation of an IDOC, see the topic on the FILEDEF command in the *Stored Procedures Reference* manual.)

If you wish to generate an IDOC synonym, you can embed commands in a procedure (focexec), as shown in the following example:

```
CREATE SYNONYM i47_idocs/MATMAS03_edioff FOR MATMAS03 IDOC  EDIOFF
DBMS SQLSAP
AT I47
DROP
END
```

Example: Creating a Synonym for an SAP Table

The following synonym was generated from a TABLE source: DD02T. The specified system name is 146. When the synonym is created, this value is reflected in the Access File.

Generated Master File DD02T.mas

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       SNAPpack generated on Tue Nov 26 17:42:06 2002
$       SAP instance: I46   SAP Release: 46C
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
FILE=DD02T, SUFFIX=SQLSAP
  REMARKS='R/3 DD: SAP table texts',$

SEGNAME=DD02T,SEGTYPE=S0,$
 FIELD=DD02T_TABNAME                        ,TABNAME ,
 A30          ,A30          ,MISSING=OFF,
 TITLE='Table',   DESC='Table name',$
 FIELD=DD02T_DDLANGUAGE                     ,DDLANGUAGE ,
 A1           ,A1           ,MISSING=OFF,
 TITLE='Lang.',   DESC='Language key',$
 FIELD=DD02T_AS4LOCAL                       ,AS4LOCAL ,
 A1           ,A1           ,MISSING=OFF,
 TITLE='Activation',   DESC='Activation status of a Repository object',$
 FIELD=DD02T_AS4VERS                        ,AS4VERS ,
 A4           ,A4           ,MISSING=OFF,
 TITLE='Version of',   DESC='Version of the entry (not used)',$
 FIELD=DD02T_DDTEXT                         ,DDTEXT ,
 A60          ,A60          ,MISSING=OFF,
 TITLE='Short text',   DESC='Short text describing R/3 Repository
objects',$
```

Generated Access File DD02T.acx

```
SYSTEM=I46,$
SEGNAME=DD02T     ,
TABLENAME=DD02T, TABLETYPE=TRANSP,
MANDT=NO,
KEYS=4
,$
```

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

SAP Table Class Support for an Individual Table

The ability of WebFOCUS to access a given table is based on its class. Class can be found in the TABCLASS field in DD02L for this particular table. The following table classes are supported.

SAP Table Type	Server Support
TRANSP	Maps to an individual SQL table (called *transparent table* by SAP). It can usually be accessed and read by the interface (see exception 1).
INTTAB	Describes a structure (called *structure* by SAP) that can be used by function modules, logical databases, and so on. There is no underlying data, and it is not supported by the interface.
CLUSTER	These tables (called *cluster table* by SAP) are supported, as long as they satisfy exception 1.
POOL	These tables (called *pooled table*) are supported, as long as they satisfy exception 1.
VIEW	Those tables are supported, as long as they satisfy exception 1, and all joined tables carry a supported table class.
APPEND	These tables are not supported.

In addition, for all supported TABLE classes, they must be composed of a set of fields that have a supported data type. Unsupported data types are listed in exception 2.

Exceptions:

1. If a field in a table is of type RAW, with a significant length (for example, field CLUSTD in table INDX), this may indicate that the field may be used for import/export using a program. Since there is no information on the underlying structure, this field can only be printed as an alphanumeric string. Its corresponding contents may not be meaningful on display.
2. Unsupported data types: VARC, PREC, LCHR, LRAW, RAWSTRING.

SAP Support for a Function Module

There are two attributes (displayed in the *Attributes* tab of the function module) to consider before using a function module:

- If the function module is not released (displayed in *general data*), it is internal to SAP, and a user should use it with extreme caution. From release to release, this function module can be removed or its behavior changed without any SAP prior notice.

- The attributes *normal function module* or *remote-enabled function module* described in the *Processing type* tab do not make a technical difference for the interface. However, if the function module is RFC-enabled, it may show that SAP has taken some steps to externalize this function module.

The interface is read-only. You should check that any function module called from the interface is read-only. There is no way for the server to check if a function module is read-only. This responsibility has to be deferred to the user.

Consideration should also be given to system security. If a function module is RFC-enabled, it does not mean it is bound to SAP security. From our own experience, only the subset of function modules with a name starting with *BAPI* may include SAP security.

1. **Create Synonym Time**

   A function module may have a set of Import/Export/Changing/Tables parameters. It may also include a set of exceptions. (See current limitations below).

   Every parameter must be prototyped to a data type that can be translated into a WebFOCUS server data type. If a parameter cannot be mapped, a warning is issued (FOC44492), and the parameter will be rejected. An example is the LOCAL_CAT import parameter in the READ_TEXT function module.

   Parameters can be prototyped as LIKE or TYPE. In case of TYPE, this type must be externalized, that is, defined outside the scope of the function group where the function module is defined. An example where the TYPE cannot be found is in the SELKZ_KUPAV table parameter in the SD_PARTNER_SELECTION function module. This parameter is typed as LV09A_TY_PRTNR_ITAB, defined in the LV09A type group. In this case, an error message will be issued (FOC44488), and the parameter will be rejected.

   The interface can handle either single fields or single-level structures, but it cannot handle compound structures, such as the IT_COMP_SIMUPARAM_RANGES import parameter in the ACEDS_ACCRUAL_FOR_ACRTYPE_CALC function module. In this case, an error message will be issued (FOC44491), and the parameter will be rejected.

   When the structure of a table parameter contains only four contiguous fields named SIGN, OPTION, contains(LOW), and contains(HIGH), this particular table is viewed as a select option table, and will be declared as such in the Access File. This, outside of single-value import, is the most efficient way to pass a set of restrictions to a function module.

2. **Limitations**

   - Currently the adapter does not support CHANGING parameters.
   - Data type prototypes must be resolved to a real unique ABAP data type. As such, ANY, ANY TABLE, STANDARD TABLE, and TABLE are not supported as data type prototypes.

- TYPE C defaults to A1.
- TYPE STRING defaults to A80. For those two types (C and STRING), it will be up to the user, with the help of the Synonym Editor, to adjust the length of defined fields to individual needs.
- Data type must be one that is supported. Unsupported data types are VARC, PREC, LCHR, LRAW, RAWSTRING.
- IMPORT parameters can be mapped to one single field (using either LIKE or TYPE).
- EXPORT parameters can be either single fields or a structure (using either LIKE or TYPE). When the EXPORT parameter is a structure defined using TYPE, there are two possibilities:
  1. The parameter is referenced in the Data Dictionary as a *structure* (for example, BAPIRET2).
  2. It is referenced as a *Table type* (for example, BAPIRET2_T).
- Even though the synonym can be created in all cases, *table type*, as an export parameter is not supported.

3. **Error/Warning Messages**

- (E) (FOC44488) DATATYPE data_type NOT FOUND FOR PARAMETER parameter_name
- (E) (FOC44489) DDIF_FIELDINFO_GET INTERNAL ERROR FOR parameter_name
- (E) (FOC44490) DDIF_FIELDINFO_GET UNDOCUMENTED ERROR FOR parameter_name
- (E) (FOC44491) NESTED STRUCTURES NOT SUPPORTED. PARAMETER parameter_name
- (E) (FOC44492) PARAMETER parameter_name DOESN'T HAVE A DATA TYPE
- (W) (FOC44493) LIKELY SELECT OPTION TABLE table_name(parameter_name) WILL BE MAPPED TO AN INTERNAL TABLE

Upon error, if the parameter on which the error occurred is optional, the synonym will still be created without the corresponding parameter. If the parameter is required, the creation of the synonym will be aborted.

Note: Properties for a given parameter are found using the DDID_FIELDINFO_GET function module. If this function module cannot retrieve properties for the parameter, the parameter is flagged with (FOC44488).

ABAP exceptions generated by a SAP function module are now trapped at TABLE run time.

The resulting error will be:

```
(FOC1695) SAP/R3 REQUEST ERROR   :
          [optional error message sent by the function module]

(FOC1736) SAP/R3 ERROR EXECUTING ABAP4 PROGRAM : function_module_name
```

SAP Data Type Support

The following table lists the SAP data types and notes how they are mapped to data types in the Master File.

SAP Data Type	SAP Length	SAP Decimals	Usage	Actual
ACCP	N	0	An	An
CHAR	N	0	An	An
CLNT	3	0	A3	A3
CUKY	N	0	An	An
CURR	n	p	P(n+2).p	P(n+1)/2
DATS * see note following chart for details	8	0	YYMD or DMYY	DATE
DEC	n	p	P(n+2).p	P(n+1)/2
FLTP ** see note following chart for details.	n	p	Dmm.eeE	D8
INT1	3	0	I3 (limited to 127)	I1
INT2	5	0	I4	I2
INT3	5	0	I4	I3

SAP Data Type	SAP Length	SAP Decimals	Usage	Actual
INT4	10	0	I10	I4
LANG	1	0	A1	A1
LCHR	N	n	not supported	
LRAW	N	N	not supported	
NUMC	N	0	An	An
PREC	N	N	not supported	
QUAN	N	P	P(n+2).p	P(n+1)/2
RAW	N	N/A	A2n	A2n
STRING *** see note following chart for details	Maximum length 2GB	N/A	A80	A80
TIMS	6	0	A6	A6 (00:00:00)
VARC	N	0	An (<=255)	An (<=255)
UNIT	3	0	A3	A3

Note:

- * **Date usage (DATS)** is dynamically determined based on the value of the country supplied in the general server configuration. Therefore, date presentations will vary by country.
- ****FLTP** The value will be expressed as a floating point number with a mantissa and an exponent.

```
Dmm.eeE
```

where:

`mm (mantissa)`

Is ee + 6. Adding 6 is the length in characters of sign_of_mantissa + mantissa_digit + character 'D' + sign_of exponent + 2-digit exponent.

`ee (exponent)`

Is the number of SAP decimals -1 (SAP accounts for the decimal point in the count).

For example, the field UMREF in the table VBAP is a floating point number with 16 digits expressed as:

ee = 16 -1 =15. mm = 15 + 6 = 21.

```
FIELDNAME=VBAP_UMREF, ALIAS=UMREF, USAGE=D21.15E, ACTUAL=D8
```

*** **STRING** defines a variable character length sequence. If you need to adjust the length of the STRING, you can do so using the Web Console or Data Management Console Synonym Editor.

SAP Open/SQL Support

The following table describes the conversion that takes places from the FOCUS statements to the SAP Open/SQL statements.

FOCUS Command	Corresponding Open/SQL statement
READLIMIT	UP TO *n* ROWS When a multiple SELECT statement is generated, READLIMIT is translated in UP TO n ROWS for each individual SELECT.
WHERE	Generally translated into a WHERE statement.
JOIN TO	For each table included in the JOIN command, embedded SELECT statements are created.
JOIN	Translated into SELECT ... SELECT SINGLE ...
NULL	The Adapter for SAP does not support the use of the reserved word 'NULL' as part of a WHERE clause. Including 'NULL' as part of a WHERE clause results in a syntax error.

FOCUS Command	Corresponding Open/SQL statement
SUM ... BY	By default, the aggregation feature is turned off. If the aggregation is turned off, individual rows are selected and aggregation and sorting is done by FOCUS. It can be turned on by issuing `ENGINE SQLSAP SET OPTIMIZATION SQL` Once the aggregation is turned on, the SUM is translated into SUM and the BY is translated into GROUP BY and ORDER BY. The aggregation feature only applies to TRANSPARENT tables.

Advanced SAP Features

This topic describes how to use security, BAPIs, joins, and tracing.

Support for User Security

The Native interface for SAP has been enhanced to handle the user ID and password in the following syntax:

```
ENGINE SQLSAP SET CONNECTION_ATTRIBUTES system/user,password:'client'
```

This feature allows you to run a secured request that involves the following:

- One or more BAPIs, as BAPIs are secured by SAP.
- SAP delivered logical databases, as the server code is secured by SAP.
- Function modules (either SAP or customer) that are fully prototyped, including the proper AUTHORITY-CHECK statements.

Using Customized Security Exits

To prevent and avoid the malicious injection of ABAP code in the context of a WebFOCUS query, the function modules provided by Information Builders (DYNAMIC_RUN, REP_CREATE, REP_GET_B_D, REP_RUN) have been enhanced to perform, by default, whitelisting of the dynamically generated ABAP4 code of legitimate, permitted commands (see the SEC_CHK function module for the list of permitted commands). Any command not whitelisted will be denied execution.

These function modules have also been enhanced to insert calls to customer-defined function modules on start (USR_EXT_ENT*) and exit (USR_EXT_EXT*). Initially, the calls are inserted to dummy versions of the function modules supplied by Information Builders. You can modify these dummy versions to implement all additions and authorizations (SAP authority checks) required by your site. These user exits will be called with the same set of parameters as the original calling function module (IMPORT/TABLES only).

Example: **Sample Customer Authority Check**

```
function /IBI/ZIBI_USR_EXT_ENT.
*"----------------------------------------------------------------------
*"*"Local Interface:
*" IMPORTING
*" VALUE(REPID) LIKE TRDIR-NAME OPTIONAL
*" VALUE(REPORT_NAME_PREFIX) LIKE RS38L-AREA OPTIONAL
*" VALUE(FUNCPOOL) LIKE RS38L-AREA OPTIONAL
*" VALUE(LDBNAME) LIKE TRDIR-LDBNAME OPTIONAL
*" VALUE(INTERFACE_MODE) LIKE DD03L-INTTYPE OPTIONAL
*" VALUE(JOB_COUNT) LIKE TRDIR-NAME OPTIONAL
*" VALUE(REPID_BD) LIKE RS38L-AREA OPTIONAL
*" TABLES
*" ITAB STRUCTURE CHAR8000 OPTIONAL
*" PROGRAM STRUCTURE ABAPTXT255 OPTIONAL
*" EXCEPTIONS
*" GENERATION_ERROR
*"----------------------------------------------------------------------

* ---------------------------------------------------
* delete a catalogued report
* ---------------------------------------------------
*{ INSERT M6DK900965 1
*
AUTHORITY-CHECK OBJECT 'S_CARRID'
ID 'CARRID' FIELD 'AA'
ID 'ACTVT' FIELD '03'.
IF sy-subrc = 4.
MESSAGE e045(sabapdocu) WITH 'AA'.
ELSEIF sy-subrc <> 0.
MESSAGE e184(sabapdocu) WITH text-010.
ENDIF.

*} INSERT
* this is now a dummied function module
ENDFUNCTION.
```

Example: Sample Exit Authorization Check for Create Synonym

The following sample exit, META_USR_SEC, is called during create synonym run-time. For every SAP Datasource type supported by the adapter, such as TABLE, BAPI, RFC FUNCTION, LDB, or QUERY, the exit can access the corresponding LOCAL INTERFACE structure that is populated with the list of create synonym candidates. The following prototype code exemplifies an AUTHORITY-CHECK on the current user SY-UNAME. The authorization has an object name of Z_*table_name*, where *table_name* is the value of 'curval', the create synonym literal parsed from the TABS structure, with an ID authorization field of A_TABNAME and contains the value of *table_name*. This dummy version should be customized with the proper authorization call for your site before being implemented.

```
FUNCTION /IBI/ZIBI_META_USR_SEC.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  TABLES
*"      NODES STRUCTURE  SNODETEXT OPTIONAL
*"      TABS STRUCTURE  DD02VV OPTIONAL
*"      BAPIS STRUCTURE  SWOTFIND OPTIONAL
*"      LDBS STRUCTURE  LDBT OPTIONAL
*"      FUGRS STRUCTURE  TLIBT OPTIONAL
*"      FUNCTIONS STRUCTURE  TFTIT OPTIONAL
*"      DEVCLASSES STRUCTURE  TDEVCVT OPTIONAL
*"      QUERY_FIELDS STRUCTURE  DFIES OPTIONAL
*"----------------------------------------------------------------------
 data: curval(30) type C.
 data: auth_object_name(80) type C.

*"----------------------------------------------------------------------
* Insert needed authorization checks here
*"----------------------------------------------------------------------
  CLEAR curval.
  LOOP AT TABS INTO curval.
*---------------------------------------------------------------
* Insert authorization check for the table name in curval here
* ---------------------------------------------------------------
     CLEAR auth_object_name.
     CONCATENATE 'Z_' curval INTO auth_object_name.
     AUTHORITY-CHECK OBJECT auth_object_name FOR USER SY-UNAME
     ID 'A_TABNAME' FIELD curval.

     IF SY-SUBRC <> 0.
        RAISE GENERATION_ERROR.
     ENDIF.
  ENDLOOP.
ENDFUNCTION.
```

BAPI Support

This release supports most read-only BAPIs, including joins, as described in the following example:

```
CREATE SYNONYM baseapp/BUS0002_GETLIST
FOR BUS0002/GETLIST
BAPI DBMS SQLSAP AT I46
END
CREATE SYNONYM baseapp/BUS0002_GETDETAIL
FOR BUS0002/GETDETAIL BAPI
DBMS SQLSAP AT I46
END
JOIN BAPI0002_COMP_CODE IN COMPANYCODE_GETLIST TO
CCGD2_COMP_CODE IN OMPANYCODE_GETDETAIL
END
TABLE FILE COMPANYCODE_GETLIST
PRINT
CCGL0_TYPE NOPRINT
BAPI0002_COMP_CODE
BAPI0002_COMP_NAME
CCGD2_CURRENCY
CCGD2_LANGU
IF BAPI0002_COMP_CODE NE '2300' OR '6000'
END
```

Join Support

The Native Interface supports all joins from and to SAP. For performance reasons, we do not recommend joining to an SAP data source from a non-SAP data source. It is more efficient to hold the keys in a sequential file, and then use the following code:

```
TABLE FILE SAP PRINT FIELDS IF KEYS IS (HOLD) END
```

Setting Up the Report Processing Mode

SAP supports many different types of processes (for example, batch, dialog (online), dynamic, print, and so on). Each WebFOCUS for SAP query creates a process within the SAP application server. This process can execute as either a dialog (online) or batch process.

These processing mode options allow you to adapt to the SAP application server configuration where there are a specific number of processes allocated for each process type per SAP application server.

You can change mode settings from the Web Console by clicking Data Adapters in the navigation pane, clicking the name of a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Setting Dialog and Batch Execution Modes

You can set execution processing to dialog (also called online) mode or batch mode.

- **Dialog (or online) processing** is the default mode for the Adapter for SAP. An SAP dialog process is suitable for a task that needs to run immediately at a higher priority. An SAP dialog process is subject to an idle timeout limit where the default value is 15 minutes. Check with your SAP Basis consultant to see if this default value has changed. Dialog process mode should be set for reports that have processing times below the idle timeout limit. If the processing time of a report exceeds the dialog process idle timeout limit, the SAP application server terminates the process and the report fails to run.

  In this mode, the report is executed online. The session is blocked until execution is completed. Execution time is bound to limits preset in the SAP system.

  The following command sets your Adapter for SAP to dialog processing mode:

  ```
  ENGINE SQLSAP SET EXECUTIONMODE ONLINE
  ```

  You would use this command if you had explicitly set batch execution mode and wanted to switch back to dialog mode within a single session on the server.

  If you have a WebFOCUS for SAP report that requires longer processing time, consider Batch processing.

- **Batch processing** is an alternative mode for the Adapter for SAP. An SAP batch process is suitable for a task that can afford to run in the background at a lower priority. This mode is for reports that require longer processing time (for example, reports that involve joining multiple tables or month end reports that need to process a lot of data). In this mode, the report will not time-out due to an execution time limit.

  The following command will set your Adapter for SAP to batch processing mode:

  ```
  ENGINE SQLSAP SET EXECUTIONMODE BATCH
  ```

To implement these settings from the Web Console, click Data Adapters in the navigation pane, click the name of a configured adapter, and choose *Change Settings* from the menu. The Change Settings pane opens. Choose *Online* or *Batch* from the Execution mode drop-down menu.

Note that you can also insert these commands directly into the server profile (for example, edasprof.prf) or included within the report procedure.

Setting Polling Intervals for Batch Execution

Before you issue any requests for batch execution, you can define how often the engine will check for the completion of these requests.

The syntax is

```
ENGINE SQLSAP SET NONBLOCK n
```

where:

n

Is the polling interval defined in number of seconds. The default is 30 seconds.

To implement this setting from the Web Console, click Data Adapters in the navigation pane, click the name of a configured adapter, and choose *Change Settings* from the menu. The Change Settings pane opens. Enter a value in the NONBLOCK field.

Note that you can also insert this command directly into the server profile (for example, edasprof.prf) or include it within the report procedure.

Producing SAP Requests

The following requests and output are examples of the capabilities of the Adapter for SAP using standard ANSI SQL code. These examples require that CREATE SYNONYM has been performed on the relevant tables from the Web Console. You can use the Web Console to navigate the SAP application hierarchy by searching the relevant tables or by doing a simple search by name.

Example: Retrieving All Records

The following syntax retrieves all records from T014:

```
SELECT * FROM T014;
```

WebFOCUS equivalent of this SQL statement is:

```
TABLE FILE T014
PRINT *
END
```

The output is:

Client	Credit Area	Currency	Update	FYI Variant	Risk Categ.	Cred. Limit	Rep. Group	All Codes
800	0001	EUR	000012			.00		
800	1000	EUR	000012	K4		.00		

Client	Credit Area	Currency	Update	FYI Variant	Risk Categ.	Cred. Limit	Rep. Group	All Codes
800	3000	USD	000012	K4		.00		
800	5000	JPY	000012	K4		.00		
800	6000	MXN	000012	K4		.00		
800	R100	EUR	000012	K4		.00		
800	R300	USD	000012	K4		.00		

Example: **Retrieving Selected Fields**

The following syntax retrieves selected fields from MARA:

```
SELECT MARA_MATNR, MARA_MTART, MARA_MATKL, MARA_WRKST, MARA_BISMT FROM
MARA WHERE MARA_MATKL='999';
```

The WebFOCUS equivalent of this SQL statement is:

```
TABLE FILE MARA
PRINT MARA_MATNR MARA_MTART MARA_MATKL MARA_WRKST MARA_BISMT
WHERE MARA_MATKL EQ '999';
END
```

The output is:

Material	Matl_type	Matl_group	Basic_matl	Matl_no_
000000000000000038	HALB	999		
AM3-GT	KMAT	999		
BG100001	HALB	999		
KR090654	HALB	999		

Example: **Joining Two Tables**

The following syntax joins the MARA database with the MARC database:

```
SELECT MARA_MATNR, MARA_WRKST,MARC_WERKS FROM MARA,MARC WHERE
MARA_MATNR=MARC_MATNR AND MARC_WERKS='5100';
```

The WebFOCUS equivalent of this SQL statement is:

```
JOIN CLEAR *
JOIN MARA_MATNR IN MARA TO ALL MARC_MATNR IN MARC AS J0
TABLE FILE MARA
PRINT MARA_MATNR MARA_WRKST MARC_WERKS
WHERE MARC_WERKS EQ '5100';
END
```

The output is:

Material	Basic_matl	Plant
NC-100-212		5100
NC-100-213		5100
NC-100-311		5100
NC-100-312		5100
NC-103-101		5100
NC-103-211		5100

Chapter 92

Using the Adapter for SAP Hana

The Adapter for SAP Hana allows applications to access specific SAP Hana data sources. The adapter converts application requests into SAP Hana calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the SAP Hana Environment

In order to use the Adapter for SAP Hana, you must install the SAP Hana driver for whichever data source you would like to access, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the SAP Hana Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some SAP Hana Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and checking that the special services section displays "JSCOM3 active".

Configuring the Adapter for SAP Hana

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a SAP Hana data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one SAP Hana data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for SAP Hana

The *SAP Hana* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the SAP Hana data source.

Driver name

Name for the SAP Hana driver:

```
com.sap.db.jdbc.Driver
```

See driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLHANA SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

SQLHANA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name.

URL

Is the URL to the location of the SAP Hana data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to data source using the SAP Hana Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

```
ENGINE SQLHANA SET CONNECTION_ATTRIBUTES CON1
  'jdbc:sap://host:port'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLHANA SET DEFAULT_CONNECTION connection
```

where:

SQLHANA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLHANA SET DEFAULT_CONNECTION SAMPLE
```

Managing SAP Hana Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLHANA to identify the Adapter for SAP Hana.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLHANA [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLHANA

Is the value for the adapter.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for SAP Hana

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support Report* on page 2315.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for SAP Hana synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLHANA ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,     DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the SAP Hana table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support Report

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the SAP Hana Environment

The Adapter for SAP Hana provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to SAP Hana.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLHANA SET TIMEOUT {nn|0}
```

where:

`SQLHANA`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a timeout occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native SAP Hana driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLHANA SET PASSRECS {ON|OFF}
```

where:

SQLHANA

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

SAP Hana Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 93

Using the Adapter for Siebel

The Adapter for Siebel allows applications to access Siebel data sources. The adapter converts data or application requests into native Siebel statements and returns optimized answer sets to the requesting program.

In this chapter:

Software Requirements for the Adapter for Siebel

Verify that the level of Java 2 SDK required for your Siebel release is installed in your environment.

To do so, issue the following command from a command prompt:

```
Java -version
```

Then, check the version returned against your Siebel documentation. If a version is not returned, contact your system administrator.

Preparing the Siebel Environment

To prepare the Siebel environment for adapter configuration, you must:

- *Setting Security* on page 2320
- *Accessing a Server* on page 2320

Setting Security

The Adapter for Siebel offers two methods of user authentication:

- **Explicit.** User IDs and passwords are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the Server's global profile, edasprof.prf, for all users.
- **Password Passthru.** User ID and password received from the client application are passed to the Siebel Server for authentication. When a client connects to the server, the user ID and password are passed to Siebel for authentication and are not authenticated by the server.

Accessing a Server

Using the standard rules for deploying the Siebel Client, the server supports connections to:

- Local Siebel database servers.
- Remote Siebel database servers.

To connect to a Siebel database server, the server must be pointing to the correct Siebel Gateway Server, Siebel Enterprise Server, Siebel Object Manager, and Siebel Server on the target machine.

Once you are connected to a Siebel database server, that server may define Siebel DATABASE LINKs that can be used to access Siebel tables on other Siebel database servers.

Preparing the Server Environment for Adapter Configuration

To prepare the server environment for adapter configuration, you must define environment variables.

In addition, if you are using Siebel 7.7 or higher and wish to take advantage of the Siebel native load balancing capability, you must add a property to the Siebel properties file and reference the associated directory in your CLASSPATH definition.

Defining Environment Variables

The Adapter for Siebel requires two server environment variables.

Procedure: How to Define Configuration Environment Variables on Windows

1. Ensure that JAVA is in the application path. For example:

   ```
   PATH=c:\jdkVersion_number\bin
   ```

2. Point directly to the JVM.dll file in your path.
3. Add the Siebel jar files to the CLASSPATH, using the following syntax:

   ```
   set CLASSPATH=jar1;jar2;[sieb_prop;]%CLASSPATH%
   ```

   where:

   `jdk`*`Version_number`*

   Begins with jdk, followed by the Java 2 SDK version installed in your environment.

   `jar1`

   Is equal to the full path specification of the SiebelJI_Common Java Class file, SiebelJI_Common.jar or Siebel.jar (for version 7.7 and higher).

   `jar2`

   Is equal to the full path specification of the SiebelJI_enu Java Class file, SiebelJI_enu.jar.

   `sieb_prop`

   Is the directory in which the Siebel.properties file resides. This entry is required for load balancing with Siebel 7.7 or higher. For related information, see *Creating a Siebel Properties File for Load Balancing* on page 2322.

Note: Contact your Siebel Administrator for access to these Siebel Java Class files. These files must reside on the same machine as the Server. The files may be copied from a machine that has a Siebel installation.

Example: Defining Configuration Environment Variables on Windows

```
set CLASSPATH=D:\IBI\SiebelJI_enu.jar;D:\IBI\Siebel.jar
```

If you are using Siebel load balancing with Siebel 7.7 or higher, the following example would apply:

```
set CLASSPATH=D:\IBI\SiebelJI_enu.jar;D:\IBI\Siebel.jar;D:\IBI
```

Procedure: How to Define Configuration Environment Variables on UNIX

1. Export the JDK_HOME variable to point to the JAVA\bin directory. For example:

   ```
   export JDK_HOME=/usr/javaVersion_number
   ```

2. Export the Shared Library variable to point to JAVA Home and the directory to LIBJVM shared library. For example:

   ```
   export LD_LIBRARY_PATH=/usr/javaVersion_number/jre/bin/classic:/usr/
      javaVersion_number/jre/bin
   ```

3. Export the Siebel jar files to your CLASSPATH, using the following syntax:

   ```
   export CLASSPATH=jar1:jar2:[sieb_prop;]$CLASSPATH
   ```

   where:

   javaVersion_number

   Begins with java, followed by the Java 2 SDK version installed in your environment.

   jar1

   Is equal to the full path specification of the SiebelJI_Common Java Class file, SiebelJI_Common.jar or Siebel.jar (for version 7.7 and higher).

   jar2

   Is equal to the full path specification of the SiebelJI_enu Java Class file, SiebelJI_enu.jar.

   sieb_prop

   Is the directory in which the Siebel.properties file resides. This entry is required for load balancing with Siebel 7.7 or higher. For related information, see *Creating a Siebel Properties File for Load Balancing* on page 2322.

Note: Contact your Siebel Administrator for access to these Siebel Java Class files. These files must reside on the same machine as the Server. The files may be copied from a machine that has a Siebel installation.

Example: Defining Configuration Environment Variables on UNIX

```
export CLASSPATH=/u1/ibi/SiebelJl_enu.jar:/u1/ibi/Siebel.jar:$CLASSPATH
```

If you are using Siebel load balancing with Siebel 7.7 or higher, the following example would apply:

```
export CLASSPATH=/u1/ibi/SiebelJl_enu.jar:/u1/ibi/Siebel.jar:/u1/ibi:$CLASSPATH
```

Reference: Creating a Siebel Properties File for Load Balancing

Beginning with Siebel 7.7. the Siebel Java Data Bean interface supports native round robin load balancing. To use this feature, you must add the following property to the siebel.properties file, assigning a list of Siebel servers with host names, port numbers, and server IDs to each virtual server. The syntax is

```
siebel.conmgr.virtualhosts=virtualserver1=sid1:host:port,sid2:host:port...;
 [virtualserver2=..]
```

where:

virtualserver1

Is the name you assign to the virtual server. You can define more than one virtual server with corresponding port, host, and sid#.

port

Is the default port number for the SCBroker (Siebel Connection Broker) component.

host

Is the machine name with Siebel Server installation.

sid#

Is the server ID.

See your Siebel documentation for additional information about Siebel load balancing.

Configuring the Adapter for Siebel

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to Siebel, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Siebel data source by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to Siebel takes place when the first query that references that connection is issued.

If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Siebel

The Siebel adapter is under the *ERP* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Gateway server: gateway port #

Siebel Gateway Server name.

For Siebel version 7.7 and higher, you must also supply the Gateway server port #.

Enterprise server

Siebel Enterprise Server name.

Object Manager

Siebel Object Manager name.

Siebel server

Siebel Server name.

Note: This parameter is not required for Siebel version 7.7 and higher.

Siebel repository

Name of the repository from which to read Siebel metadata.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Username by which you are known to Siebel. This field only displays when the security mode of the server is non-trusted.

Password

Password associated with the user name. This field only displays when the security mode of the server is non-trusted.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Explicit. The user ID and password are explicitly stated in SET CONNECTION_ATTRIBUTES commands. You can include these commands in the Server's global profile, edasprof.prf, for all users.

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES connection_name
/userid,password;:'Gateway/Enterprise/Obj Mngr/Siebel'
```

Password Passthru. The user ID and password received from the client application are passed to the Siebel Server for authentication. When a client connects to the server, the user ID and password are passed to Siebel for authentication and are not authenticated by the server.

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES connection_name :'Gateway[:gateway
port_#]/Enterprise/Obj Mngr/Siebel'
```

where:

`SIBIN`

Indicates the adapter.

`connection_name`

Is a logical name used to identify this particular set of connection attributes. In the Access File, it becomes the CONNECTION= attribute.

`userid`

Is the primary authorization ID by which you are known to Siebel.

`password`

Is the password associated with the primary authorization ID.

`'Gateway/Enterprise/Obj Mngr/Siebel'`

Is the Siebel connection string. This information must be supplied.

`Gateway[:gateway port #]`

Is the Siebel Gateway Server name. For Siebel version 7.7 and higher, you must also supply the Gateway server port #.

Enterprise

Is the Siebel Enterprise Server name.

Obj Mngr

Is the Siebel Object Manager name.

Siebel

Is the Siebel Server name. This parameter is not required for Siebel version 7.7 and higher.

Example: Declaring Connection Attributes

Explicit authentication:

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES CON1
/USER_ID,PASSWORD;:'ARIBA01/ARIBA01/SCCObjMgr/SiebelSrv'
```

Password Passthru authentication:

```
ENGINE SIBIN SET CONNECTION_ATTRIBUTES CON2
:'devportal2/siebel/EAIObjMgr/devportal2'
```

Managing Siebel Metadata

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Siebel data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Business Component (BC) that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Siebel

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Filter by Business Objects

Select this option to filter the Business Objects for which you wish to create synonyms.

When you filter by name, the Business Object Name input box is displayed. Enter a string for filtering the Business Objects, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select Business Objects which begin with the letters ABC; %ABC to select Business Objects which end with the letters ABC; %ABC% to select Business Objects which contain the letters ABC at the beginning, middle, or end.

Select BOs button

Click this button to display the Business Objects that meet the specified criteria.

Filter Business Components

Select this option to filter the Business Components for which you wish to create synonyms. When you choose to filter, the Business Component Name input box is displayed.

Enter a string for filtering the Business Component, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter ABC% to select Business Component which begin with the letters ABC; %ABC to select Business Component which end with the letters ABC; %ABC% to select Business Component which contain the letters ABC at the beginning, middle, or end.

Select Company Object

Select the business object for which you want to see business components.

Select BCs button

Click this button to display the *Select Business Components*. All Business Components that meet the specified criteria appear.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Select Business Object and Component

- To select all Business Objects/Business Components in the list, select the check box to the left of the Default Synonym Name column heading.
- To select specific Business Objects/Business Components, select the corresponding check boxes.

Example: Creating a Synonym

To generate a synonym for the ACCOUNT Siebel Business Object and Siebel Business Component, enter the following information on the Create Synonym panes of the Web Console or the Data Management Console:

1. Click the Filter Business Objects check box, enter *ACCOUNT* in the Name field, and click the *Select BOs* button.
2. On the second Create Synonym pane, click the Filter Business Components check box, enter *ACCOUNT*, and click the *Select BCs* button.
3. Select *Account* in the Default Synonym name column.
4. Click *Create Synonym*. The synonym is created and added under the specified application directory (baseapp is the default).

A status window displays the message: *All Synonyms Created Successfully*

5. From the message window, click *Applications* on the menu bar.
6. Open the *baseapp* application folder in the navigation pane and click the synonym *account*.
7. Choose *Edit as Text* from the menu to view the generated Master File, then choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File account.mas

```
FILENAME=ACCOUNT, SUFFIX=SIBIN   , $
  SEGMENT=ACCOUNT, SEGTYPE=S0, $
    FIELDNAME=VISIBILITY_MODE, ALIAS=VISIBILITY_MODE, USAGE=A15, ACTUAL=A15,
      ACCEPT='SalesRep' OR 'Manager' OR 'Personal' OR 'All' OR 'Organization' OR
      'Contact' OR 'Group' OR 'Catalog' OR 'SubOrganization', $
    FIELDNAME=ACCOUNT_COMPETITORS, ALIAS='Account Competitors', USAGE=A500,
      ACTUAL=A500, MISSING=ON, $
    FIELDNAME=ACCOUNT_CONDITION, ALIAS='Account Condition', USAGE=A500, ACTUAL=A500,
      MISSING=ON, $
    FIELDNAME=ACCOUNT_MARKETS, ALIAS='Account Markets', USAGE=A500, ACTUAL=A500,
      MISSING=ON, $
    FIELDNAME=ACCOUNT_ORGANIZATION_INTEGRATION_ID,
      ALIAS='Account Organization Integration Id', USAGE=A30, ACTUAL=A30,
      MISSING=ON, $
    FIELDNAME=ACCOUNT_PRODUCTS, ALIAS='Account Products', USAGE=A500, ACTUAL=A500,
      MISSING=ON, $
    FIELDNAME=ACCOUNT_ROLE, ALIAS='Account Role', USAGE=A30, ACTUAL=A30, MISSING=ON,
$
.
.
.
    FIELDNAME=MISSION, ALIAS=Mission, USAGE=A500, ACTUAL=A500,MISSING=ON, $
    FIELDNAME=NAME, ALIAS=Name, USAGE=A100, ACTUAL=A100, MISSING=ON, $
    FIELDNAME=NAME_AND_LOCATION, ALIAS='Name and Location', USAGE=A255, ACTUAL=A255,
      MISSING=ON, $
    FIELDNAME=NOT_MANAGER_FLAG, ALIAS='Not Manager Flag', USAGE=A255, ACTUAL=A255,
      MISSING=ON, $
.
.
.
```

Generated Access File account.acx

```
SEGNAME=ACCOUNT, CONNECTION=SIB1, SEGNAME_MVG=Account,
  REPOSITORY=Siebel Repository, BS_OBJECT=Account, BS_COMPONENT=Account,$
SEGNAME=CREDIT_CONTROL_AREA_CODE, SEGNAME_MVG=Credit Control Area Code, $
SEGNAME=SYNONYM, SEGNAME_MVG=Synonym, $
SEGNAME=BILL_TO_POSTAL_CODE, SEGNAME_MVG=Bill To Postal Code, $
SEGNAME=BILL_TO_FIRST_NAME, SEGNAME_MVG=Bill To First Name, $
SEGNAME=BILL_ADDRESS_FLAG, SEGNAME_MVG=Bill Address Flag, $
SEGNAME=DEDUP_KEY_UPDATE, SEGNAME_MVG=DeDup Key Update, $
SEGNAME=INDUSTRY, SEGNAME_MVG=Industry, $
SEGNAME=BACK_OFFICE_DISTRIBUTION_CHANNEL,
  SEGNAME_MVG=Back Office Distribution Channel, $
SEGNAME=TYPE_MVF, SEGNAME_MVG=Type MVF, $
SEGNAME=ROW_STATUS, SEGNAME_MVG=Row Status, $
SEGNAME=TERRITORY, SEGNAME_MVG=Territory, $
SEGNAME=AGREEMENT_END_DATE, SEGNAME_MVG=Agreement End Date, $
SEGNAME=SHIP_TO_POSTAL_CODE, SEGNAME_MVG=Ship To Postal Code, $
SEGNAME=SHIP_TO_FIRST_NAME, SEGNAME_MVG=Ship To First Name, $
SEGNAME=TERRITORY_0015, SEGNAME_MVG=Territory, $
```

The Master File root segment describes the Siebel Business Component. All scalar fields of the Business Component are described as fields in the root segment. All Multi Value fields (MVG) of the Business Component are described as descendant segments of the root. The name of the segment representing an MVG is derived from the Business Component Multi Value field name. All other Multi Value fields related to the same Multi Value Link are not shown in the Master File. The MVG segment contains references to the appropriate fields in the linked Business Component instead. All scalar fields of the MVG are described as fields in the MVG segment. All MVG fields of an MVG become descendant segments of the segment representing this parent MVG.

Master File field aliases represent real Siebel Business Component field names. Long names (between 67 and 75 characters) are stored in the Access File. Corresponding Master File fields must have no aliases. Similarly, long segment names are stored in the Access File as well.

A special virtual field VISIBILITY_MODE is added to the root segment to enable the client to set the visibility type for the Business Component. Available visibility modes are enumerated in the ACCEPT parameter for this field. Siebel Versions 6 and 7 support different sets of visibility modes, which is reflected in the ACCEPT list.

The visibility modes are:

Siebel v7: SalesRep, Manager, Personal, All, Organization, Contact, Group, Catalog, and SubOrganization.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Syntax: How to Set Visibility Mode

```
IF VISIBILITY_MODE EQ 'All'
```

This syntax tells Siebel to look at the user ID referenced in the .prf file and according to that user's ID authorization privileges, Siebel returns a view of ALL for the requested output.

If no visibility mode is set in the report, it defaults to the personal visibility mode.

Example: Using Visibility Mode in a Report Request

The following request prints the NAME field where the VISIBILITY MODE is set to All.

```
TABLE FILE ACCOUNT
 PRINT NAME VISIBILITY_MODE
 WHERE ( VISIBILITY_MODE EQ 'All' )
 AND ( NAME LIKE '%WLI%' ) ;
END
```

The output is:

```
 NAME                                                VISIBILITY_MODE
 ----                                                ---------------
 Yelena WLI 81 0129 iway55                           All
 Yelena WLI 81 0129 iway55 JCA2                      All
 Yelena WLI 81 IDE                                   All
```

Example: Using Visibility Mode in a SQL Request

The follow selects NAME and VISIBILITY_MODE based on VISIBILITY MODE and wildcards in NAME.

```
SQL
 SELECT NAME, VISIBILITY_MODE FROM ACCOUNT
 WHERE VISIBILITY_MODE = 'All' AND NAME LIKE '%WLI%'
END
```

The output is:

```
NAME                                    VISIBILITY_MODE
 ----                                    ---------------
 Yelena WLI 81 0129 iway55               All
 Yelena WLI 81 0129 iway55 JCA2          All
 Yelena WLI 81 IDE                       All
```

Reference: Access File Attributes

Keyword	Description
CONNECTION=*connection_name*	Indicates access to specified Siebel Server. Defaulted to the default connection.
SEGNAME	Name of the corresponding segment in the Master File.
SEGNAME_LONG	Long (1 to 75 characters) Siebel segment name.
BS_OBJECT	Siebel Business Object name.
BS_COMPONENT	Siebel Business Component name.
REPOSITORY	Siebel Repository.

Data Type Support

All Siebel scalar, alphanumeric, and currency fields are treated as alphanumeric fields. All Siebel date and integer fields are supported.

Siebel Optimization Settings

The Adapter for Siebel supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

The block size for a SELECT request applies to TABLE FILE requests, MATCH requests, and DIRECT SQL SELECT statements.

Syntax: How to Specify Block Size for Array Retrieval

```
ENGINE SIBIN SET FETCHSIZE n
```

where:

SIBIN

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of records to be passed from the physical layer of the adapter, written in Java, to the logical layer, implemented in C. Accepted values are 1 to 5000.

1 indicates non-blocked communication. 1 is the default value.

This value is associated with all data retrieval operations and applicable to all segments.

Example: Testing Retrieval Performance

The following procedure uses the FETCHSIZE command to test retrieval performance. It shows the difference in milliseconds based on the FETCHSIZE.

```
-DEFAULT &FS = 50
SET PANEL=999
ENGINE SIBIN SET FETCHSIZE 1
-RUN
-SET &&STARTCPU = &FOCCPU;
-RUN
TABLE FILE ACCOUNT
SUM NAME ACCOUNT_COMPETITORS
ON TABLE HOLD
END
-RUN
-SET &DIFFCPU = &FOCCPU - &&STARTCPU ;
-TYPE1 CPU FetchSize= 1 : &DIFFCPU MS
-RUN
ENGINE SIBIN SET FETCHSIZE &FS
-RUN
-SET &&STARTCPU = &FOCCPU;
-RUN
TABLE FILE ACCOUNT
SUM NAME ACCOUNT_COMPETITORS
ON TABLE HOLD
END
-RUN
-SET &DIFFCPU = &FOCCPU - &&STARTCPU ;
-TYPE1 CPU FetchSize= &FS : &DIFFCPU MS
```

Chapter 94

Using the Adapter for Slack

Using the Adapter for Slack enables WebFOCUS to integrate with the Slack business messaging application. WebFOCUS can retrieve and post Slack messages.

In this chapter:

Preparing the Slack Environment

The Slack Adapter requires a Slack App created in the Slack API environment prior to adapter configuration (*https://api.slack.com/apps*).

The main purpose of the Slack App is to configure OAuth20 Security used by the Slack API.

For instructions, right-click the adapter and click *Prerequisites*.

Configuring the Adapter for Slack

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish. The Adapter for Slack is in the Social Media group.

Prior to configuring the adapter, you must create an application in Slack.

Procedure: How to Configure the Adapter for Slack

1. From the Web Console sidebar, click *Connect to Data*.
2. Click the *New Datasource* button on the menu bar, and find the adapter on the page.

   You can select a category of adapter from the drop-down list, or use the search option (magnifying glass) to search for specific characters. The Adapter for Slack is in the Social Media category.
3. Right-click the adapter name and click *Configure*.

The *Add Slack to Configuration* page opens on which you can configure a connection. You can add additional connections by right-clicking the adapter on the Configured Adapters list and clicking *Add Connection*.

Reference: Connection Attributes for Slack

The Adapter for Slack is under Social Media Adapters. Enter or select values for the following connection parameters to add a connection to the Adapter for Slack.

Connection Name

Is the logical name used to identify this particular set of connection attributes. The default is CON01.

Client ID

Is the client ID defined in the Slack application. To obtain this value, do the following:

1. Go to *https://api.slack.com/apps*.
2. Click the App Name for the previously created Slack application.

Client Secret

Is the Client Secret defined in the Slack application. To obtain this value, do the following:

1. Go to *https://api.slack.com/apps*.
2. Click the App Name for the previously created Slack application.

Workspace ID

Is a unique identifier for the Slack Workspace. To obtain this value, do the following:

1. Sign on to the desired Workspace within Slack by going to *https://slack.com*.
2. Once in the desired Slack Workspace, view the page source and search for "team_id".

   The Workspace ID is the value next to "team_id", for example, "team_id":"TEK21GK63".

Workspace Name

Is the name of the Workspace. The Workspace name is populated automatically after a successful *Get Access Token* request.

Change OAuth Scope

The following scopes will always be requested as part of Get Access Token:

- ❑ channels:history
- ❑ channels:read
- ❑ users:read

- chat:write:user

Change OAuth Scope allows you to select or deselect optional scopes, for example, admin.

If the optional scopes have changed, *Get Access Token* must be rerun.

Access Token

Is the value that identifies the user on whose behalf your Slack application is acting. Click *Get Access Token* to obtain this token.

In order for *Get Access Token* to complete successfully, the host name used to bring up the Reporting Server Web Console must match the host name set up for the Redirect URI in the Slack application.

If an issue arises while obtaining the Access Token, ensure that you are signed out of all Slack Workspaces, and clear the browser cache, including cookies.

Select profile

Is the profile (server, group, user) in which to store these connection attributes. The default is the server profile, edasprof.prf.

Advanced Connection Options

Base URL

Is the URL of the Slack API request. The default value is *https://slack.com/api*.

Authorization URL

Is the URL used for OAuth Authorization to Slack. The default value is *https://slack.com/oauth/authorize*.

Token URL

Is the URL used for obtaining an Access Token to Slack. The default value is *https://slack.com/oauth/oauth.access*.

Advanced HTTP Connection Options

PROXY Server IP Address

Is the IP address of the proxy server.

PROXY Port

Is the port number on which the proxy server listens. The default port number is 80.

PROXY HTTPS Relative Path

When checked, the REST request will send the relative path rather than the absolute path when a proxy server is configured.

SSL Certificate

Is the location of a locally-stored user-provided server x.509 certificates file for SSL authentication. For trusted authentication, the trusted certificate file (trustedcertfile) points to a file of CA certificates in PEM format, as illustrated in the following syntax:

```
-----BEGIN CERTIFICATE-----
... (CA certificate in base64 encoding) ...
-----END CERTIFICATE-----
```

SSL Mutual Authentication

When checked, Mutual Authentication is enabled.

SSL Certificate Type

Select one of the following certificate types:

- **Trusted.** A trusted certificate file can contain several CA certificates. You can add text before, between, and after any certificate which is typically done to provide descriptions of each certificate.
- **Non-trusted.** Adds the parameters Key file, Pass phrase, and Label to the configuration pane. Provide an initial path browse value to the SSL Certificate field before browsing. For example:

  ```
  C:\, /abc/abc/...
  ```

SSL Certificate Key File

Is the private key used for creating the client X.509 certificate in PEM format. This option is used together with a certificate for a non-trusted connection. Provide an initial value to the SSL certificate key file field before browsing. For example:

```
C:\, /abc/abc/...
```

SSL Certificate Pass Phrase

Is the password used to unlock the key file. The value is needed only if the key file is encrypted.

SSL Certificate Label

Identifies a certificate in the file, if the file contains more than one certificate. If the label contains spaces, the label must be enclosed in double-quotes. For example:

```
"xxx yyy"
```

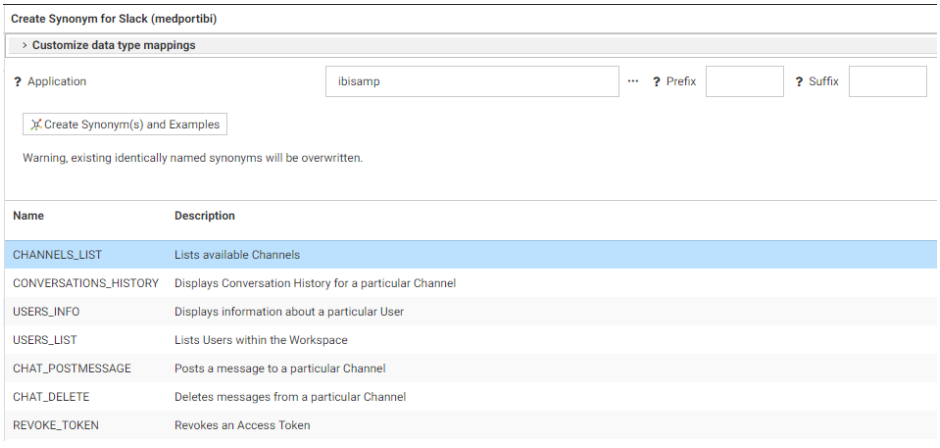
Creating Slack Metadata and Sample Reports

Create Synonym for the Adapter for Sleck creates the metadata used for WebFOCUS reporting and DataMigrator ETL flows. It also creates sample WebFOCUS reports.

Procedure: How to Create Slack Metadata and Sample Reports

1. Right-click a configured Slack connection, and click *Show DBMS objects*.

   The Create Synonym for Slack page opens, as shown in the following image.



2. Enter an application name in the Application field. or click the ellipsis button to the right of the field to select an application where the metadata and sample reports are to be stored.
3. Click *Create Synonym(s) and Examples*.

   The metadata and examples are generated in the specified application directory.

Example: Sample Slack Synonym and Report

The following is the Master File generated for the conversations_history object.

```
FILENAME=M6ILO, SUFFIX=SLACK    , REMARKS='Displays Conversation History
for a particular Channel', $
  SEGMENT=M6ILO, SEGTYPE=S0,
$
   GROUP=HEADER, ALIAS=Header, ELEMENTS=3,
$
    FIELDNAME=CHANNEL, ALIAS=channel, USAGE=A30, ACTUAL=A30,
              ACCESS_PROPERTY=(NEED_VALUE), $
       FIELDNAME=OLDEST, ALIAS=oldest, USAGE=A17, ACTUAL=A17,
                ACCESS_PROPERTY=(NEED_VALUE), $
       FIELDNAME=LATEST, ALIAS=latest, USAGE=A17, ACTUAL=A17,
              ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
              ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=JSON    , PARENT=M6ILO,
POSITION=__RESPONSE, $
    FIELDNAME=JSON_DUMMY_EL, ALIAS=JSON_DUMMY_EL, USAGE=A1, ACTUAL=A1,
              ACCESS_PROPERTY=(INTERNAL),
              PROPERTY=ELEMENT,
$
    FIELDNAME=OK, ALIAS=ok, USAGE=A55, ACTUAL=A55,
              REFERENCE=JSON_DUMMY_EL, PROPERTY=ELEMENT,
$
    FIELDNAME=HAS_MORE, ALIAS=has_more, USAGE=A55, ACTUAL=A55,
              REFERENCE=JSON_DUMMY_EL, PROPERTY=ELEMENT,
$
    FIELDNAME=PIN_COUNT, ALIAS=pin_count, USAGE=P33, ACTUAL=A33,
              REFERENCE=JSON_DUMMY_EL, PROPERTY=ELEMENT,  $
```

```
 SEGMENT=MESSAGES, SEGTYPE=S0, PARENT=RESPONSE,
$
    FIELDNAME=MESSAGES, ALIAS=messages, USAGE=A1, ACTUAL=A1,
              ACCESS_PROPERTY=(INTERNAL),
              REFERENCE=JSON_DUMMY_EL, PROPERTY=ELEMENT,
$
    FIELDNAME=CLIENT_MSG_ID, ALIAS=client_msg_id, USAGE=A55, ACTUAL=A55,
              REFERENCE=MESSAGES, PROPERTY=ELEMENT,
$
    FIELDNAME=TYPE, ALIAS=type, USAGE=A55, ACTUAL=A55,
             REFERENCE=MESSAGES, PROPERTY=ELEMENT,
$
    FIELDNAME=TEXT1, ALIAS=text, USAGE=A2000, ACTUAL=A2000,
              REFERENCE=MESSAGES, PROPERTY=ELEMENT,
$
    FIELDNAME=USER1, ALIAS=user, USAGE=A55, ACTUAL=A55,
              REFERENCE=MESSAGES, PROPERTY=ELEMENT,
$
    FIELDNAME=TS, ALIAS=ts, USAGE=P17.6, ACTUAL=A17,
              REFERENCE=MESSAGES, PROPERTY=ELEMENT,
$
    FIELDNAME=SUBTYPE, ALIAS=subtype, USAGE=A55, ACTUAL=A55,
              REFERENCE=MESSAGES, PROPERTY=ELEMENT,  $
```

The following is the Access File generated for the conversations_history object.

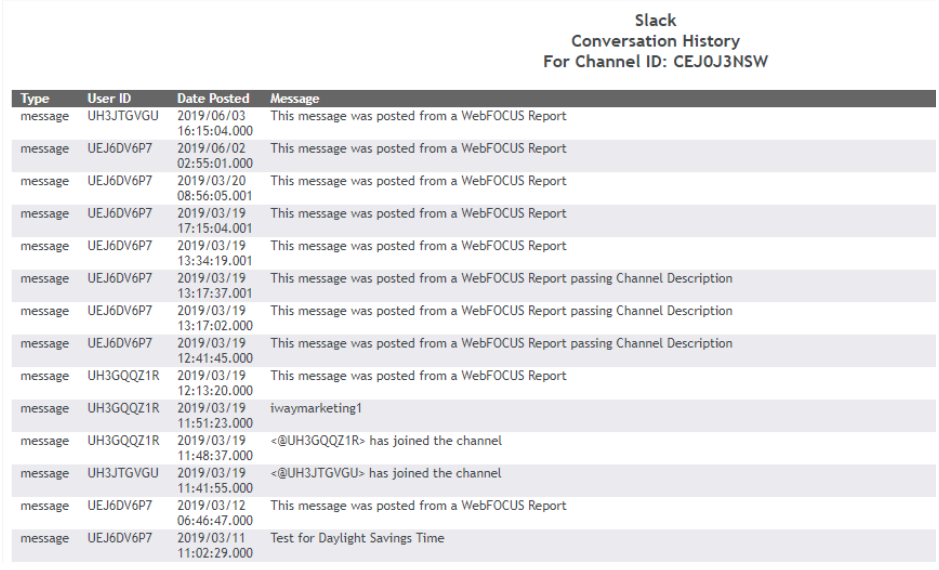
```
SEGNAME=M6ILO,
  CONNECTION=medportibi,
  OBJECT=conversations.history,
  HEADER=HEADER,
  SERVICETYPE=REST,
  HTTPMETHOD=GET,
  RESTRESPONSE=JSON, $
```

The following is a report request that uses this synonym.

```
-
*DM_JOB_TYPE=0

-*DM_REQ_DESC=Displays Conversation History for a particular
Channel
-
*

-DEFAULT
&CHANNELID='CEJ0J3NSW'
-
*

-*Determine Timezone Hour
Offset
-SET &NOW =
DT_CURRENT_DATETIME(MILLISECOND);

-SET &NOWZ =
HGETZ(8,'HYYMDs');
-SET &ANOW =
DT_FORMAT(&NOW,'HYYMDs');
-SET &ANOWZ =
DT_FORMAT(&NOWZ,'HYYMDs');
-SET &OFFSET =
DTDIFF(&NOWZ,&NOW,HOUR);
-
*

DEFINE FILE ibisamp/
CONVERSATIONS_HISTORY
BASEDATE_HYYMDS/HYYMDS=HINPUT(19, '1970-01-01 00:00:00', 8,
'HYYMDS');
TS_HYYMDS/HYYMDs=IF TS GT 0 THEN DTADD(HADD(BASEDATE_HYYMDS, 'MILLISECOND',
TS*1000 , 8, 'HYYMDS'),HOUR, &OFFSET*(-1)) ELSE 0;
END

-*
```

```
TABLE FILE ibisamp/
CONVERSATIONS_HISTORY
PRINT

     CONVERSATIONS_HISTORY.TYPE AS
'Type'
     CONVERSATIONS_HISTORY.USER1 AS 'User
ID'
     TS_HYYMDS AS 'Date
Posted'
     CONVERSATIONS_HISTORY.TEXT1 AS
'Message'
WHERE CHANNEL EQ '&CHANNELID.Channel
ID.'
HEADING

"Slack"

"Conversation
History"
"For Channel ID:
&CHANNELID"
ON TABLE SET PAGE-NUM
NOLEAD
ON TABLE
NOTOTAL
ON TABLE PCHOLD FORMAT
HTML
ON TABLE SET HTMLCSS
ON
ON TABLE SET STYLE
*
     INCLUDE =
endeflt,
$

ENDSTYLE

END
```

Partial output is shown in the following image.



Slack
Conversation History
For Channel ID: CEJ0J3NSW

Type	User ID	Date Posted	Message
message	UH3JTGVGU	2019/06/03 16:15:04.000	This message was posted from a WebFOCUS Report
message	UEJ6DV6P7	2019/06/02 02:55:01.000	This message was posted from a WebFOCUS Report
message	UEJ6DV6P7	2019/03/20 08:56:05.001	This message was posted from a WebFOCUS Report
message	UEJ6DV6P7	2019/03/19 17:15:04.001	This message was posted from a WebFOCUS Report
message	UEJ6DV6P7	2019/03/19 13:34:19.001	This message was posted from a WebFOCUS Report
message	UEJ6DV6P7	2019/03/19 13:17:37.001	This message was posted from a WebFOCUS Report passing Channel Description
message	UEJ6DV6P7	2019/03/19 13:17:02.000	This message was posted from a WebFOCUS Report passing Channel Description
message	UEJ6DV6P7	2019/03/19 12:41:45.000	This message was posted from a WebFOCUS Report passing Channel Description
message	UH3GQQZ1R	2019/03/19 12:13:20.000	This message was posted from a WebFOCUS Report
message	UH3GQQZ1R	2019/03/19 11:51:23.000	iwaymarketing1
message	UH3GQQZ1R	2019/03/19 11:48:37.000	<@UH3GQQZ1R> has joined the channel
message	UH3JTGVGU	2019/03/19 11:41:55.000	<@UH3JTGVGU> has joined the channel
message	UEJ6DV6P7	2019/03/12 06:46:47.000	This message was posted from a WebFOCUS Report
message	UEJ6DV6P7	2019/03/11 11:02:29.000	Test for Daylight Savings Time

Chapter 95 Using the Adapter for Snowflake Cloud Data Warehouse

The Adapter for Snowflake Cloud Data Warehouse allows applications to access Snowflake cloud data sources. The adapter converts data or application requests into native Snowflake SQL statements and returns optimized answer sets to the requesting program.

Two types of the adapter, ODBC and JDBC, are available. Only one type can be configured on a server instance.

In this release, only the JDBC type of the adapter is certified.

In this chapter:

Preparing the Snowflake JDBC Environment

Snowflake provides a JDBC type 4 driver that requires Java 1.7 or higher and must be installed in a 64-bit environment.

Before starting the server, make sure that path to the directory where the Snowflake driver is installed is included in IBI_CLASSPATH.

For example (on Linux):

```
export IBI_CLSSPATH=/qas/snowflake/snowflake-jdbc-3.6.3.jar
```

Preparing the Snowflake Cloud Data Warehouse ODBC Environment

The Snowflake Cloud Data Warehouse ODBC environment is available on Windows and Linux Red Hat versions 7 and 6.

Download and install the Snowflake ODBC driver for your operating environment. If your operating environment is Linux, you may need to install the Driver Manager for Linux, if this was not previously done.

Configuring the Adapter for Snowflake Cloud Data Warehouse (JDBC)

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Snowflake Cloud Data Warehouse Adapter (JDBC)

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: **Snowflake Cloud Data Warehouse Adapter Configuration Settings (JDBC)**

The Adapter for Snowflake Cloud Data Warehouse is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the Snowflake data source.

The basic syntax is:

```
jdbc:snowflake://Server/?warehouse=&|db=&|schema=
```

You can add any additional connection parameters, such as tracing=off, by first adding the delimiter characters ampersand followed by a vertical bar (&|) to the end of the string.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Driver Name

Is the name of the JDBC driver. For example:

```
net.snowflake.client.jdbc.SnowflakeDriver
```

`IBI_CLASSPATH`

Is the jar class path for the driver. For example:

```
/qas/snowflake/snowflake-jdbc-3.6.3.jar
```

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Always use the characters &| (ampersand followed by the vertical bar) as the parameter separator in the URL for the JDBC type of Snowflake adapter.

Extended Bulk Load functionality is supported with the JDBC type of Snowflake Adapter and does not require any additional configuration steps.

Configuring the Adapter for Snowflake Cloud Data Warehouse (ODBC)

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Snowflake Cloud Data Warehouse Adapter (ODBC)

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.

In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Snowflake Cloud Data Warehouse Adapter Configuration Settings (ODBC)

The Adapter for Snowflake Cloud Data Warehouse is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

DSN-less

A DSN-less connection is an alternative method for connecting to Snowflake Cloud Data Warehouse. When this parameter is selected, the adapter is configured by entering values for the Snowflake server, warehouse, default database, and default schema instead of a Data Source Name (DSN).

Server

Specifies the full domain name for your account (provided by Snowflake). For example:

```
ar59294.us-east-1.snowflakecomputing.com
```

Warehouse

Specifies the default warehouse to use for a session initiated by the driver. For example:

```
QAWH
```

Default Database

Specifies the default database to use for a session initiated by the driver. For example:

```
QATST
```

Default Schema

Specifies the default schema to use for a session initiated by the driver. For example:

```
R729999D
```

DSN

Is a valid Snowflake Data Source Name (DSN). There is no default DSN, you must enter a value. The DSN name should match the User, System, or File DSN configured in the ODBC Administrator on Windows, or the DSN entry in the $HOME/.odbc.ini file on Linux.

Security

There are three methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Additional connection string keywords (optional)

Is used to add options to the connection string. For example, the tracing=0 parameter is required when no Snowflake ODBC traces are needed.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Creating Synonyms With Snowflake Cloud Data Warehouse

Synonyms define unique names, or aliases, for each Snowflake Cloud Data Warehouse table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Snowflake Cloud Data Warehouse table.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Chapter 96

Using the Adapter for SQLBase

The Adapter for SQLBase allows applications to access SQLBase data sources. The adapter converts application requests into SQLBase calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the SQLBase Environment

In order to use the Adapter for SQLBase, you must install the SQLBase driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

The Adapter for SQLBase only supports access to U.S. databases.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the SQLBase Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server's EDAPRINT. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for SQLBase

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a SQLBase data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one SQLBase data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for SQLBase

The *SQLBase* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the SQLBase data source.

Driver name

Name for the SQLBase JDBC driver.

See SQLBase documentation for the specific driver release you are using.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLBAS SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLBAS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the SQLBase data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a books database using the SQLBase Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

Note: Consult vendor documentation for the exact name, port, and path.

```
ENGINE SQLBAS SET CONNECTION_ATTRIBUTES CON1
  'jdbc:sqlbase://edared30:2155/island'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLBAS SET DEFAULT_CONNECTION connection
```

where:

SQLBAS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects CON1 as the default connection.

```
ENGINE SQLBAS SET DEFAULT_CONNECTION CON1
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLBAS SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLBAS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing SQLBase Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLBAS to identify the Adapter for SQLBase.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLBAS [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLBAS

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using SQLBase, you must locally install the RDBMS SQLBase Driver.

The Server can access third-party database tables across the network SQLBase. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note: If you are creating a synonym for a SQLBase data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for SQLBase

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.

- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  `/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE`

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 2368.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for SQLBase synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLBAS ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the SQLBase table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the SQLBase Environment

The Adapter for SQLBase provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to SQLBase.

Syntax: **How to Issue the TIMEOUT Command**

```
ENGINE SQLBAS SET TIMEOUT {nn|0}
```

where:

`SQLBAS`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Setting*s from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLBAS SET PASSRECS {ON|OFF}
```

where:

SQLBAS

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

SQLBase Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 97

Using the Adapter for Stratio Crossdata

The Adapter for Crossdata provides for analysis of both structured and complex data. Crossdata is a fast and general engine for large-scale data processing.

In this chapter:

Introducing the Adapter for Stratio Crossdata

Stratio Crossdata is a fast and general engine for large-scale data processing.

Crossdata is a Big Data platform from Stratio (www.stratio.com) that is based on Apache Spark and HDFS without any dependency on Apache Hadoop.

The Adapter for Stratio Crossdata is JDBC based. It uses the Stratio JDBC driver. The DataMigrator or WebFOCUS server can be run on any platform (including Windows) that connects to the server where Crossdata is running.

Preparing the Crossdata Environment

The following components are needed to use the adapter:

- **Java.** To use this JDBC-based adapter, you must have Java installed. Crossdata requires Version 1.8 or later. Java can be downloaded from *http://www.java.com*.

  The location of Java must be specified in an environment variable.

  If you are using Linux, add a line to your profile with the location where Java is installed. For example:

  ```
  export JAVA_HOME=/usr/lib/jvm/jre-1.8.0
  ```

If you have JDK installed:

```
export JAVA_HOME=/usr/lib/jvm/jdk-1.8.0
```

If you are using Windows, right-click *Computer* and click *Properties*. Then click *Advanced System Settings* and click *Environment Variables*. Add the locations to your PATH variable. For example:

```
C:\Program Files\Java\jdk8\bin\server;C:\Program Files\Java\jdk8\bin;
```

- **JDBC Drivers.** The Stratio Crossdata JDBC driver requires one jar file, which is included with Stratio Crossdata. If you are installing the server on the same system where Crossdata is installed, you can point to the jar file as described in the next section. If you are installing the server on some other system, copy the file to a location of your choice.

Procedure: How to Configure the Java CLASSPATH

The location of the Stratio Crossdata jar file must be specified to the server. If you are running the server on the same system as the Stratio Crossdata server, you can specify its location. If you are running the server on another system, copy the file listed below to some location on your system and specify its location.

This can be done in the system CLASSPATH or in the DataMigrator or WebFOCUS Reporting Server IBI_CLASSPATH variable as follows:

1. From the Web Console sidebar, select *Workspace*

   or

   From the Data Management Console, expand the *Workspace folder*.

2. Expand the *Java Services* folder. Right-click *DEFAULT* and click *Properties*.

   The Java Services Configuration page opens.

3. Expand *Class path*.

   In the IBI_CLASSPATH box, enter the full location of the Stratio Crossdata .jar file shown below.

   If you are installing the adapter on a different system than where Stratio Crossdata is installed, copy the jar file to a location on that system.

   Enter the location on your system that matches your installation.

   ```
   stratio-crossdata-jdbcversion.jar
   ```

Configuring the Adapter for Stratio Crossdata

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Procedure: How to Configure the Adapter for Stratio Crossdata

You can configure the adapter from either the Web Console or the Data Management Console.

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click *Stratio Crossdata* and click *Configure JDBC*.

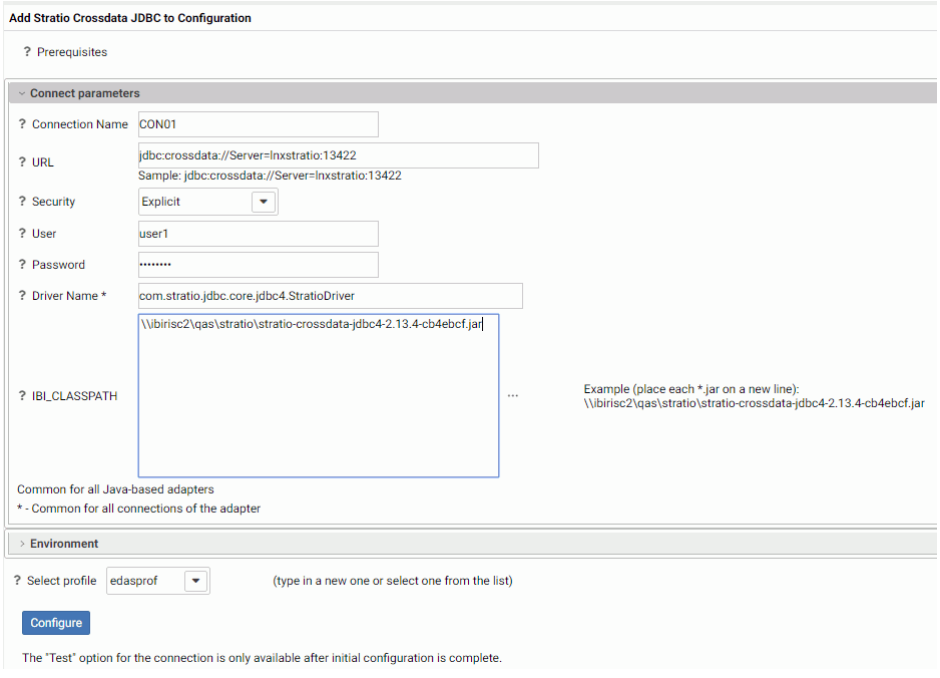
5. In the URL box, type the URL used to connect to your Crossdata server. For more information, see *Adapter for Stratio Crossdata Configuration Settings* on page 2374.

6. In the Driver Name box, type the following driver name:

   ```
   com.stratio.jdbc.core.jdbc4.StratioDriver
   ```

7. Select the security type. If you are using Explicit, type your user ID and password.

The following image shows an example of the configuration settings used:



8. Select *edasprof* from the Select profile drop-down menu to add this connection for all users, or select a user profile.
9. Click *Configure*.
10. Click *Test*. You should see a list of data sources on your server.

Reference: Adapter for Stratio Crossdata Configuration Settings

The Adapter for Stratio Crossdata is under the SQL group folder.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Is the URL to the location of the data source. For example:

```
jdbc:crossdata://Server=lnxstratio:13422;LogLevel=0;LogPath=.
```

where:

`server`
: Is the DNS name or IP address of the system where the Crossdata server is running. If it is on the same system, localhost can be used.

`default`
: Is the name of the default database to connect to.

`13422`
: Is the default port number the Crossdata Server is listening on if not specified when the Crossdata server is started.

Driver Name

Is the name of the JDBC driver, com.stratio.jdbc.core.jdbc4.StratioDriver.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. This value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions when setting values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar). When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Troubleshooting

If the server is unable to configure the connection, an error message is displayed. An example of the first line in the error message is shown below, where *nnnn* is the message number returned.

```
(FOC1400) SQLCODE IS -1 (HEX: FFFFFFFF) XOPEN: nnnn
```

Some common errors messages are:

```
[00000] JDBFOC>> connectx(): java.lang.UnsupportedClassVersionErr : or: org/
stratio/crossdata/driver : Unsupported major.minor version 51.0
```

The adapter requires Java 1.8 or later and your JAVA_HOME points to Java 1.6.

```
(FOC1500)  :  ERROR: ncjInit failed. failed to connect to java server: JSS
(FOC1500)  :   . JSCOM3 listener may be down - see edaprint.log for details
```

The server could not find Java. To see where it was looking, review the edaprint.log and set JAVA_HOME to the actual location. Finally, stop and restart your server.

```
(FOC1260)  (-1) [00000] JDBFOC>> connectx():
java.lang.ClassNotFoundException:
   com.stratio.jdbc.core.jdbc4.StratioDriver
(FOC1260) Check for correct JDBC driver name and environment variables.
(FOC1260) JDBC driver name is com.stratio.jdbc.core.jdbc4.StratioDriver
(FOC1263) THE CURRENT ENVIRONMENT VARIABLES FOR SUFFIX SQLCRD ARE :
(FOC1260) IBI_CLASSPATH : ...
```

The JDBC driver name specified cannot be found in the jar files specified in IBI_CLASSPATH or CLASSPATH. The names of the jar files are either not specified, or if specified, do not exist in that location.

```
[08S0] Could not establish connection to jdbc:crossdata://Server=hostname:
13422
:
java.net.UnknownHostException: hostname
```

The server hostname could not be reached on the network. Check that the name is spelled correctly and that the system is running. Check that you can ping the server.

```
[08S01] Could not establish connection to localhost:13422/default:
: java.net.ConnectException: Connection refused
```

The Stratio Crossdata server is not listening on the specified port. Start the server if it is not running, and check that the port number is correct.

Creating Synonyms With Stratio Crossdata

Synonyms define unique names, or aliases, for each Crossdata table that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given Crossdata table.

Procedure: How to Create a Synonym With Stratio Crossdata

To create a synonym, you must have previously configured the adapter and a connection.

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The Status pane indicates that the synonym was created successfully. The synonym is created and added under the specified application directory.

Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Using Direct Pass-through With Stratio Crossdata

Direct Pass-through commands, such as SHOW TABLES and DESCRIBE tablename, do not display any results.

This can be avoided by adding a comment:

```
ENGINE SQLCRD
/*QUERY*/ SHOW TABLES ;
END
```

Loading Data into Stratio Crossdata Using DataMigrator

DataMigrator can be used to load data from any accessible data source into Crossdata and create metadata. Currently, the only load type supported is Extended Bulk Load. The exception are files stored as ORC (Optimized Row Columnar) format, for which Insert/Update can be used.

The table should be stored as TEXTFILE (default), ORC, or Parquet.

For more information, see the *DataMigrator User's Guide*.

Chapter 98

Using the Adapter for Supra

The Adapter for Supra allows applications to access Supra data sources. The adapter converts application requests into native Supra statements and returns optimized answer sets to the requesting application.

The following topics discuss how the Adapter for Supra is configured and how data is mapped to its corresponding server counterparts in the z/OS environment.

In this chapter:

Preparing the Supra Environment

Prior to starting the server, it is necessary to perform the following pre-configuration procedures when preparing the environment on z/OS.

Syntax: How to Customize the LINKEDIT JCL

After executing ISETUP, the JCL will be generated to LINKEDIT the supplied Adapter for Supra module with the Supra stub called CSVILUV. The CSVILUV module is supplied by the Supra vendor, Cincom, and usually resides in the Supra LINKLIB library.

This required step allows the Adapter for Supra module to communicate with the Supra Central PDM.

Modify the following LINKEDIT JCL to conform to site standards and submit it for execution. The JCL listed below displays the GENESUPR JCL that is available at *qualif*.HOME.DATA:

```
//*        Job Card Goes Here
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose: Linkedit SUPRA API routines into Supra Adapter           *
//*         This is required if you run applications that connect   *
//*         to SUPRA database.                                       *
//*                                                                  *
//*Substitutions:                                                   *
//*         In the SET statements, supply the values for 'supra',   *
//*         'target' and 'home' according to description.           *
//*                                                                  *
//*    target library must be placed ahead of home library in        *
//*    STEPLIB concatenation of IRUNJCL.                             *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//        SET  SUPRA=<SUPRA LINKLIB object library>
//        SET  TARGET=<Library to receive relink modules>
//        SET  HOME=<Installation HOME.LOAD library>
//LKED    EXEC PGM=IEWL,PARM='MAP,XREF'
//SUPRA   DD   DISP=SHR,DSN=&SUPRA
//SYSLMIN DD   DISP=SHR,DSN=&HOME
//SYSLMOD DD   DISP=SHR,DSN=&TARGET
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE SUPRA(CSVILUV)
  INCLUDE SYSLMIN(FSP000)
  MODE AMODE(64),RMODE(24)
  ENTRY FSP000
  NAME FSP000(R)
/*
```

where:

`SET SUPRA=<SUPRA LINKLIB object library>`
: Specifies the fully-qualified name of the SUPRA linklib library that contains the CSVILUV module.

`SET TARGET=<Library to receive relink modules>`
: Specifies the fully-qualified name of the load library receiving the relink module.

`SET HOME=<Installation HOME.LOAD library>`
: Specifies the fully-qualified name of the server installation HOME.LOAD load library.

Syntax: How to Customize the Server IRUNJCL JCL

If, during ISETUP installation, you selected *Yes* for Supra Adapter, the IRUNJCL will already contain the STEPLIB allocations for Supra. If you missed that at install time, you can manually add these allocations to the server IRUNJCL.

The Supra DBMS has three load libraries: LINKLIB, INTERFLM, and ENVLIB. These load libraries must be concatenated to ddname STEPLIB.

Syntax:

How to Install the Adapter for Supra Security Exit (Optional)

The Adapter for Supra has a security exit, FSPEXT, that allows the user to obtain the account name and account password from a user-defined source. The account name and account password are then passed to the adapter and used for login to the Central PDM.

When you issue a request against the Supra database, the adapter calls the FSPEXT function with standard IBM calling conventions, and passes it these parameters:

USR	Passed to function (Server Logon User ID)	8 Bytes Alpha
ACCNT	Returned from function	16 Bytes Alpha
ACCNTP	Returned from function	16 Bytes Alpha

The function returns ACCNT and ACCNTP to the adapter, which uses them as parameters to the sign-on call it issues. The adapter calls the user exit if the password is not available from other sources (for example, the SET command or an Access File). The exit is loaded as a module using a LOAD macro from ddname USERLIB, TASKLIB, or STEPLIB, in that order. It is called in AMODE 31. The syntax for the call to the FSPEXT function is:

```
CALL FSPEXT(USR,ACCNT,ACCNTP)
```

Procedure:

How to Utilize the Security Exit

To utilize the security exit, perform the following steps.

1. Write the function using COBOL or Assembler. The function name must be FSPEXT.

An example of the FSPEXT function written in Assembler follows.

```
* SAMPLE EXIT FOR Supra ADAPTER

      EJECT
FSPEXT  CSECT
      USING FSPEXT,R15       TEMPORARY ADDRESSABILITY
      B     PROC             JUMP AROUND THE ID INFORMATION
* ID INFORMATION
      DC    CL8'FSPEXT '     PROG NAME
         PRINT NOGEN
PROC STM    R14,R12,12(R13)
      LR    R12,R15
      DROP  R15
      USING FSPEXT,R12
      ST    R13,SAVAR+4
      MVC   8(4,R13),=A(SAVAR)
      LA    R13,SAVAR
*
*     INSERT THE CODE TO FETCH THE ACCOUNT /PASSWORD
       L     R2,4(R1)
       MVC   0(16,R2),=CL16'EXITACCOUNT     '
       L     R2,8(R1)
       MVC   0(16,R2),=CL16'EXITPASS        '
*
       L     R13,SAVAR+4
       LM    R14,R12,12(R13)
       BR    R14
*************************************************************
       SAVAR  DS    18F
        R1    EQU   1
        R2    EQU   2
        R9    EQU   9
        R10   EQU   10
        R11   EQU   11
        R12   EQU   12
        R13   EQU   13
        R14   EQU   14
        R15   EQU   15
*
         END
```

2. LINKEDIT the object module into the '*qualif*.HOME.LOAD' library with AMODE(31).

For example:

```
//LINK     EXEC PGM=IEWL,PARM='LET,NCAL,SIZE=(1024K),LIST'
//OBJLIB   DD   DSN=objlib,DISP=SHR
//SYSLMOD  DD   DSN=qualif.HOME.LOAD,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(10,1))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE OBJLIB(FSPEXT)
  ENTRY FSPEXT
  MODE AMODE(31)
  NAME FSPEXT(R)
/*
//
```

where:

objlib
: Is the fully-qualified name of the library containing the FSPEXT object code.

qualif. HOME.LOAD
: Is the adapter load library.

3. If you LINKEDIT the object module into a load library other than '*qualif*.HOME.LOAD', concatenate the load library from Step 2 to the STEPLIB DD in your server JCL.

Configuring the Adapter for Supra

You can configure the adapter from either the Web Console or the Data Management Console.

Note: The SUPRA adapter is under the **DBMS** group folder.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Declaring Connection Attributes

In order to access data on the Supra Central Physical Data Manager (PDM), you must have a valid user name and password. You can issue the user name and password to the Central PDM in several ways:

- Hard code the user name and password in the Access File with the USER= and PASSWORD= attributes.
- Issue the Adapter for Supra SET commands from any supported server profile. The syntax is:

  ```
  ENGINE SUPRA SET USER userid
  ENGINE SUPRA SET PASSWORD password
  ```

- Invoke the security exit. The security exit, described in *How to Install the Adapter for Supra Security Exit (Optional)* on page 2381, allows you to code a procedure that returns the user name and password to the Adapter for Supra which are used when the signon call is issued to the Central PDM.

The adapter ascertains the user name and password with the following steps:

1. It checks the Access File.
2. If the user name and password are not coded in the Access File, the adapter checks whether the user issued the SET commands to set the user name and password.
3. If the adapter has not resolved the user name and password using the Access File or SET commands, it invokes the security exit.

Supra Overview and Mapping Considerations

This topic explains how the Adapter for Supra enables you to describe and report from Supra database views. You should become familiar with these topics because most of the concepts affect the way Supra files are described to the server.

Each Supra database can be described as a segment in a Master File. You can describe multiple views in a single Master File, if the key of one view is a field in the other view. This embedded cross-reference is described with the KEYFLD and IXFLD parameters in the Access File.

Reference: Supra RDM

A Supra database is a collection of one or more views. The Supra Relational Data Manager (RDM) provides access to physical fields from one or more files in a flat, two-dimensional format. A set of field values is a row. A Supra database consists of one or more views, each of which consists of one or more rows. You can define a subset of rows or reorder the columns according to your needs. This subset of data is called a *user view*. A column is the smallest logical unit of data a program or user can request.

After creating the view, the database administrator (DBA) defines it in the directory. The DBA can also designate one or more columns as keys to the view. The keys can be used to locate a specific set of rows.

In summary, the terms essential for understanding the data model are described in the following table.

Term	Description
View	Set of one or more rows defined by the Database Administrator (DBA).
User View	Subset of a view, which may consist of all or part of the view.
Row	Set of one or more related data items stored in computer memory.
Column	In a row, a specified area used for a particular category of data.
Value	Quantity assigned to a constant, variable, parameter, or symbol.
Key	One or more data items, the contents of which identify the type or location of a row, or the ordering of data.

Keys can appear anywhere in the view. The DBA can define different types of keys:

- **Simple Unique Key.** The simplest view has one unique key. This key enables you to select and retrieve data. A unique key must have a valid, non-null value.
- **Compound Unique Key.** The DBA designates several columns as unique logical keys, and the combination of the key values is unique. That is, an *and* connection between the columns is implied. For example, to check customer orders for a certain part number, you would use a view with both customer number and part number as key values. RDM retrieves the specific customer number and part number combination, if it is present.
- **Simple Non-unique Key.** This kind of key allows more than one row to contain the same value in a key column. An example is a customer file where a list of notes or comments about each customer is stored, and they are not dated or distinct. In this case, the customer number could be a simple non-unique key. When the program does its first GET using a customer number, it retrieves the first comment for this customer. A subsequent GET retrieves the second, and so on. After RDM retrieves the last comment for that customer, it will reach a boundary condition and return a NOT FOUND condition to the program.

You can access a set of rows by assigning values to the keys of the view (if there are any). The DBA determines which columns are keys and defines them on the directory. You can use the keys to locate a specific record or to perform a generic read. You can also access a set of rows sequentially, by not supplying any values for the keys.

Reference: Mapping Supra Views and Fields

To map a Supra structure to the server you must create two files, the Master File and the Access File.

Supra constructs correspond to the following Master File elements.

- One view or user view can be defined as a single segment.
- A column from a Supra view or user view becomes a field.

  The Master File should include only needed columns from the Supra view. Performance may be improved if you do not specify all of the columns defined in the view. Even though it is possible to define the columns in an arbitrary order, rearranging key columns may adversely affect performance.
- For each field, the ALIAS must be defined and must be the Supra view column name.

The Access File contains Supra DBMS-specific information.

Managing Supra Metadata

This topic explains how the Adapter for Supra enables you to describe and report from Supra database views. You should become familiar with these topics because most of the concepts affect the way Supra files are described to the server.

When the server accesses a data source, it needs information on how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Supra data types.

Creating a Supra Master File

A Master File is a set of comma-delimited records. Each record is terminated with a comma and dollar sign (,$). The syntax for describing a Supra database in a Master File is similar to that of any other Master File and uses the same attributes (keyword=value pairs). A Master File can contain a File record, Segment records, and Field records. The attributes are described in the following sections.

Reference: File Keywords

FILENAME

Is an arbitrary name up to eight characters in length.

SUFFIX

Is SUPRA when accessing RDM or PDM data using the Supra RDM and identifies the adapter.

Reference: Segment Keywords

SEGNAME

Is a name for the segment, up to 64 characters in length.

SEGTYPE

Is the segment type. S*n* means that the first *n* fields are components of the key, in the order listed. This information is used only if the Access file is missing, or if no value is specified for the KEYNAME attribute of the Access File. If the two values disagree, the value specified in the Access File is retained. A message is displayed only if FSTRACE is allocated to the screen or to a file.

PARENT

Is the segment name of the parent. If you describe several Supra views or user views as a hierarchical structure in one Master File, you must define one view as the root segment and all other views as descendant segments. Each descendant segment must include the PARENT attribute to identify its parent in the hierarchy. You must also identify the shared fields for each pair of segments by specifying the KEYFLD and IXFLD attributes in the Access File.

Reference: Field Keywords

FIELDNAME

Is an arbitrary name, up to 66 characters in length.

ALIAS

Is the name of the view column, as defined to Supra. The total length must not exceed 66 characters.

USAGE

Is the display format for the field in report output. USAGE format values are A (Alpha), I (Integer), F (Floating point), D (Decimal), and P (Packed). Each format must have an associated length. F, D, and P formats can also have a specification for the number of decimal places to display, for example, D12.2 specifies a total length of 12, with a period and two decimal places.

ACTUAL

Specifies the actual Supra format (that is, the format defined by the DBA in the view definition that an application program would use for accessing the data).

The following are examples of Supra formats and their corresponding ACTUAL formats.

- Supra B 1, 2, 4, and 8 (with decimal) fields can be defined as F or D.
- Supra C 1-32,767 fields can be defined as A (the maximum length of an alphanumeric field is 256 characters).
- Supra F 4, 8, and 16 fields can be defined as I.
- Supra P 1-16 (with decimal) fields can be defined as P.
- Supra Z 1-18 (with decimal) fields can be defined as Z.

The ACTUAL format does not specify the number of decimal places in the number. The USAGE format defines the number of decimal places. It must be large enough to accommodate all of the digits in the number, the decimal point, and a possible minus sign. For example, if the Supra format is P5 with 2 decimal places, you can use the following USAGE and ACTUAL specifications in the Master File:

```
ACTUAL=P5, USAGE=P8.2
```

Note: You can describe multiple views in one Master File. If they have shared fields (fields with the same value in both views), one view is described as the parent segment, and the other views are described as the child segments.

Creating a Supra Access File

The Access File provides the information necessary for selecting the appropriate Supra database, view, and access strategy (that is, the linking of items between different tables).

A logical record in the Access File consists of a list of attributes (keyword = value pairs) separated by commas and terminated by a comma and dollar sign (,$). The list is freeform and can span several lines. The keywords can be specified in any order.

The Access File contains two types of logical records, HEADER and SEGMENT.

Reference: HEADER Logical Record

USER
Is the Supra user name. If omitted, it defaults to *FOCUS*.

PASSWORD
Is the password associated with the Supra user. If omitted, it defaults to *FOCUS*.

Reference: SEGMENT Logical Record

SEGNAME
Is the segment name from the Master File.

VIEW
Is the name of the Supra view. If omitted, it defaults to SEGNAME.

USERVIEW
Is the Supra user view corresponding to the segment identified by SEGNAME. If omitted, it defaults to VIEW.

KEYNAME

Is a composite of at most nine field names that constitute the key of the view. The field names are concatenated with the slash symbol (/), without intervening blanks. The keyword value can span more than one line. If omitted, the first *n* fields, as specified by the SEGTYPE keyword in the Master File, are assumed.

KEYFLD

Is a field name in the parent segment or a defined field that establishes the embedded cross-reference between segments. It is mandatory in all but the root segment. Its value is a composite of a maximum of nine field names concatenated with the slash symbol (/), without blank spaces. The value can span more than one line.

IXFLD

Is a field name in the descendant segment that establishes the embedded cross-reference between segments. It is mandatory for all but the root segment. The IXFLD values should match the KEYFLD values of the parent segment. The keyword value consists of a maximum of nine field names concatenated with the slash symbol (/), without blanks. The value can span more than one line.

If your request references fields from two segments defined in the same Master File, the two views are automatically cross-referenced. The fields that implement the cross-reference are specified in the Access File with the KEYFLD and IXFLD attributes. The selection of the shared fields is transparent to you.

The following rules apply to KEYFLD and IXFLD.

- As KEYFLD, you can specify a:
  - Simple field.
  - List of fields.
- As IXFLD, you can specify a:
  - Simple field.
  - List of fields.

In all cases, the length and the format of KEYFLD and IXFLD must be consistent (a list of fields may be joined to a simple field and vice versa, as long as their formats are alphanumeric).

Sample Master and Access File

The following is a sample Master File that describes two Supra views.

```
FILE=DTCCORP1               ,SUFFIX=SUPRA,$
SEGNAME=CORPDESC,SEGTYPE=S,$
  FIELDNAME=DELETE_FLAG     ,ALIAS=DFLAG.1   , USAGE=A1    ,ACTUAL=A1   ,$
 GROUP=CONTRL_KEY           ,ALIAS=CNTRL.KEY,USAGE=A23   ,ACTUAL=A23 ,$
  FIELDNAME=CORP            ,ALIAS=CNTRL.1   ,USAGE=A3     ,ACTUAL=A3   ,$
  FIELDNAME=ACCOUNT         ,ALIAS=CNTRL.2   ,USAGE=A10    ,ACTUAL=A10 ,$
  FIELDNAME=COST_CENTR      ,ALIAS=CNTRL.3   ,USAGE=A10    ,ACTUAL=A10 ,$
 FIELDNAME=LRECL            ,ALIAS=LRECL.1   ,USAGE=P4     ,ACTUAL=P3   ,$
 FIELDNAME=ACCT_TYPE        ,ALIAS=ACDSC.1   ,USAGE=A1     ,ACTUAL=A1   ,$
 FIELDNAME=ACCT_DESC        ,ALIAS=ACDSC.2   ,USAGE=A20    ,ACTUAL=A20 ,$
 FIELDNAME=FILLER           ,ALIAS=ACDSC.3   ,USAGE=A32    ,ACTUAL=A32 ,$
SEGNAME=CORPOWNR,SEGTYPE=S,PARENT=CORPDESC,$
  FIELDNAME=DELETE_FLAG     ,ALIAS=EFLAG.1   ,USAGE=A1     ,ACTUAL=A1   ,$
 GROUP=DESC_KEY             ,ALIAS=DTKEY.KEY,USAGE=A23  ,ACTUAL=A23 ,$
  FIELDNAME=CORP            ,ALIAS=DTKEY.1   ,USAGE=A3     ,ACTUAL=A3   ,$
  FIELDNAME=ACCOUNT         ,ALIAS=DTKEY.2   ,USAGE=A10    ,ACTUAL=A10 ,$
  FIELDNAME=COST_CENTR      ,ALIAS=DTKEY.3   ,USAGE=A10    ,ACTUAL=A10 ,$
 FIELDNAME=LRECL            ,ALIAS=LRECL.1   ,USAGE=P4     ,ACTUAL=P3   ,$
 FIELDNAME=ACCT_TYPE        ,ALIAS=OWNER.1   ,USAGE=A1     ,ACTUAL=A1   ,$
 FIELDNAME=ACCT_OWNER       ,ALIAS=OWNER.2   ,USAGE=A20    ,ACTUAL=A20 ,$
 FIELDNAME=FILLER           ,ALIAS=OWNER.3   ,USAGE=A32    ,ACTUAL=A32 ,$
```

The following is the corresponding Access File.

```
USER=CINCOM,PASSWORD=CINCOM,$
SEGNAME=CORPDESC,VIEW=V01,KEYNAME=CNTRL.KEY,$
SEGNAME=CORPOWNR,VIEW=V02,KEYNAME=DTKEY.KEY,
    KEYFLD=CNTRL.KEY,IXFLD=DTKEY.KEY,$
```

Adapter Tracing

To activate the Adapter for Supra Tracing Facility, use the Web Console. From the Web console main page, click the User menu, and then *Enable Traces*.

The default traces will include the Adapter for Supra information.

Chapter 99

Using the Adapter for Sybase

The Adapter for Sybase allows applications to access Sybase data sources. Both Sybase ASE and Sybase IQ servers are supported.

The adapter converts application requests into native Sybase statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Sybase Environment (OCS)

The Adapter for Sybase requires the installation of the Sybase Open Client (CT-Library). The Sybase client allows you to connect to a local or remote Sybase database server after the platform-specific environment variables are set. After installation, the client must be configured to allow you to connect to a local or remote Sybase database server.

Identifying the Location of the Interfaces File

If you are using Sybase inherent distributed access, you must set the SYBASE environment variable to identify the directory where the *interfaces* file resides. The Windows equivalent of the UNIX *interfaces* file is the sql.ini file.

The *interfaces* file is required for both local and remote access. If you do not set the SYBASE environment variable, Sybase searches /etc/passwd for the login directory of the user named Sybase and accesses the *interfaces* file in that directory.

Note that the UNIX PATH must include the location of the Sybase executables. For example, specify the PATH entry as follows for Sybase:

```
SYBASE=/usr/sybase
PATH=$PATH:$SYBASE/bin
export SYBASE PATH
```

You can export the SYBASE environment variable from the UNIX shell or in the UNIX profile of the user's machine that starts the server.

For information about other environment variables needed for Sybase executables and components, see the Sybase installation and configuration documentation.

Specifying the Sybase Server Name

Use the DSQUERY environment variable to specify the Sybase Server name. The server uses this value only if:

- You do not issue the SET CONNECTION_ATTRIBUTES command in the global server profile (edasprof.prf).
- You do not specify the CONNECTION= attribute in the Access File.

Procedure: How to Specify the Sybase Server Name

```
DSQUERY=server
export DSQUERY
```

where:

server

Is the name of the Sybase database server.

Accessing a Remote Sybase Server

Using the standard rules for deploying the Sybase Client, the server supports connections to:

- Local Sybase servers.
- Remote Sybase servers. To connect to a remote Sybase server, the *interfaces* file (*sql.ini* on Windows) must contain an entry pointing to the target machine and the listening process must be running on the target machine.

New entries in the *interfaces* file may be made using the Sybase tool DSEDIT. Entries may also be made automatically using the sybinstall facility. For details, see the Sybase documentation.

Unicode Support

For ASE, beginning with Sybase ASE version 15.0, the adapter supports Unicode data in Sybase ASE databases that have been created with the UTF-8 character set.

For IQ, beginning with Sybase IQ version 12.7, the adapter supports Unicode data in Sybase IQ databases that have been created with the UTF-8 character set.

You must set the LANG environment variable in the edastart file or in a separate shell file before starting the server. For example, for American English, you would export the following variable:

```
export LANG=EN_US.UTF-8
```

To take advantage of adapter support for Unicode data, the Reporting Server must be configured with UTF-8 code page 65001.

XA Support(ASE)

Read/write applications accessing Sybase data sources are able to perform transactions managed in XA-compliant mode.

To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web console configuration functions. Using Transaction Coordination Mode guarantees the integrity of data modification on all of the involved DBMSs and protects part of the data modifications from being committed on one DBMS and terminated on another.

For complete documentation on XA compliance, see *XA Support* on page 2689.

Preparing the Sybase Environment (JDBC)

In order to use the JDBC Adapter for Sybase, you must install the JDBC driver and its CLASSPATH value before server startup and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the JDBC Environment on Windows and UNIX

1. Identify the location of the JDBC Driverfiles by adding them to the CLASSPATH environment variable before server startup.

   For example, to set the JDBC Driver files to /qas/sybase_ASE/jconn4.jar on UNIX, issue the following commands:

   ```
   CLASSPATH=/qas/sybase_ASE/jconn4.jar;$CLASSPATH
   export CLASSPATH
   ```

   On UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

On Windows, you would issue the following commands:

```
set CLASSPATH=%c:\qas\sybase_ASE\jconn4.jar:
%CLASSPATH%
```

Note that if you run the server workspace as a Windows service, you must set CLASSPATH as a system-wide environment variable.

2. Start, or Restart, the server.

Configuring the Adapter for Sybase

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to the Sybase database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Sybase database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Sybase Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Sybase (OCS)

The Sybase adapter is under *SQL* on the Available drop-down list.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Server

Name of the Sybase server to access. It must match an entry in the Sybase interfaces file.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.

- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Reference: Connection Attributes for Sybase (JDBC)

The Sybase adapter is under *SQL* on the Available drop-down list.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the JDBC data source.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Driver name

Name of the JDBC driver.

For information, see the driver documentation for the specific release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names that will be available for Java Services. This value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

Explicit authentication. The user ID and password are explicitly specified for each connection and passed to Sybase, at connection time, for authentication.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES connection server/userid,password
```

Password passthru authentication. The user ID and password are explicitly specified for each connection and passed to Sybase, at connection time, for authentication.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES connection server/
```

where:

connection

Is the logical name used to identify this particular set of connection attributes.

SQLSYB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

server

Is the name of the Sybase server you wish to access.

userid

Is the primary authorization ID by which you are known to Sybase.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES CON01 SAMPLESERVER/MYUSER,PASS
```

The following SET CONNECTION_ATTRIBUTES command connects to the Sybase database server named SAMPLESERVER using Password Passthru authentication:

```
ENGINE SQLSYB SET CONNECTION_ATTRIBUTES CON01 SAMPLESERVER/
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLSYB SET DEFAULT_CONNECTION connection
```

where:

SQLSYB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

  ```
  FOC1671, Command out of sequence.
  ```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

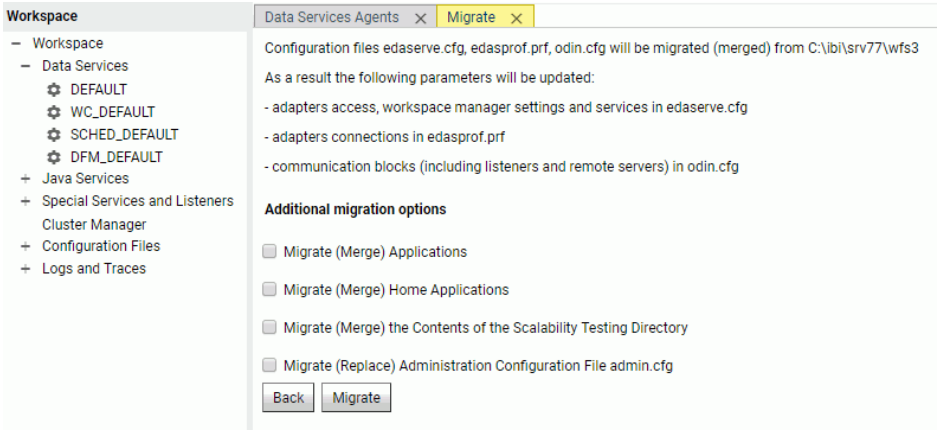
```
ENGINE SQLSYB SET DEFAULT_CONNECTION SAMPLE
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to a newer release. To access this option, select *Workspace* from the menu bar, then select *Migrate* from the ribbon. This is the recommended approach.

On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click *Continue*. Select additional migration options, as shown in the following image, and click *Migrate*



If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior to Version 7 Release 6.1, the connection was defined as:

  ```
  ENGINE SQLSYB SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

  ```
  ENGINE SQLSYB SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
  ```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLSYB SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLSYB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Sybase Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Sybase data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Sybase table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Sybase

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Database selection

To specify a database from which you can select a table or other object, do one of the following:

- Check *Use current database* to use the database that has been set as the default database.
- Select a database from the *Select database* drop-down list, which lists all databases in the current DBMS instance.

  Before selecting a database, if *Use current database* is checked, uncheck it.

This option applies to Sybase ASE; it does not apply to Sybase IQ.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Sybase Data Type Support* on page 2411.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Sybase synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File

```
FILENAME=NF29004, SUFFIX=SQLSYB  , $
  SEGMENT=NF29004, SEGTYPE=S0, $
    FIELDNAME=DIVISION4, ALIAS=DIVISION4, USAGE=I11, ACTUAL=I4, $
    FIELDNAME=DIVISION_NA4, ALIAS=DIVISION_NA4, USAGE=A25, ACTUAL=A25,
      MISSING=ON, $
    FIELDNAME=DIVISION_HE4, ALIAS=DIVISION_HE4, USAGE=I11, ACTUAL=I4,
      MISSING=ON, $
```

Generated Access File

```
SEGNAME=NF29004, TABLENAME=qatst.R729999D.NF29004,
  CONNECTION=CON01,KEYS=0, $
```

Reference: Mapping Sybase IQ Table Comments Into a Synonym

When a synonym is created, the adapter picks up comments from a Sybase IQ database table and places them in the synonym:

- Sybase IQ *Comment on Table* maps to REMARKS in the Master File.
- Sybase IQ *Comment on Field* maps to DESCRIPTION in the Master File.

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Identifies the Sybase table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local Sybase database server. Absence of the CONNECTION attribute indicates access to the default database server.

Keyword	Description
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	Set to Yes to enable auto-incrementing.
START	Initial value in the incrementing sequence.
INCREMENT	Increment interval.

Keyword	Description
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Accessing Different Databases on the Same Sybase Server

When the server connects to Sybase, it uses a primary authorization ID. This ID is associated with a default database on the server. To access tables in another database, you must specify a fully qualified name as follows:

database.owner.table

This fully qualified name can be used in the SQL request. It *must* be used in the CREATE SYNONYM command for a table in a database other than the default associated with the connected primary authorization ID. If you are allowing access to multiple databases on the same Sybase Server, the user ID specified in the SET CONNECTION_ATTRIBUTES command must have authority to access the database, as well as the tables within the database. This can be achieved by using the Sybase stored procedure sp_adduser and the SQL GRANT statement. For more information, see the *Sybase SQL Reference* manual.

Sybase Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Controlling the Mapping of Variable-Length Data Types

For Sybase ASE, the SET parameter VARCHAR controls the mapping of the Sybase data types VARCHAR, VARCHAR2, and NVARCHAR2. By default, the server maps these data types as variable character (A*n*V).

For Sybase IQ, the SET parameter VARCHAR controls the mapping of the Sybase data types VARCHAR and VARBINARY. By default, the server maps these data types as variable character (A*n*V).

Reference: Sybase ASE: Mapping of Variable-Length Data Types

The following table lists data type mappings based on the value of VARCHAR:

		VARCHAR ON		VARCHAR OFF	
Sybase ASE Data Type	**Remarks**				
VARCHAR (*n*)	*n* is an integer between 1 and 255	A*n*V	A*n*V	A*n*	A*n*
NVARCHAR (*n*)	*n* is an integer between 1 and 255	A*n*V	A*n*V	A*n*	A*n*
VARBINARY (*n*)	*n* is an integer between 1 and 255*m* = 2 * *n*	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types (Sybase ASE)

```
ENGINE SQLSYB SET VARCHAR {ON|OFF}
```

where:

`SQLSYB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Sybase data types VARCHAR, VARCHAR2, and NVARCHAR2 as variable-length alphanumeric (A*n*V). This is required for Unicode environments. ON is the default value.

`OFF`

Maps the Sybase data types VARCHAR, VARCHAR2, and NVARCHAR2 as alphanumeric (A).

Reference: Sybase IQ: Mapping of Variable-Length Data Types

The following table lists data type mappings based on the value of VARCHAR:

Sybase IQ Data Type	Remarks	VARCHAR ON		VARCHAR OFF	
VARCHAR (*n*)	*n* is an integer between 1 and 255	A*n*V	A*n*V	A*n*	A*n*
VARBINARY (*n*)	*n* is an integer between 1 and 255 *m* = 2 * *n*	A*m*V	A*m*V	A*m*	A*m*

Syntax: How to Control the Mapping of Variable-Length Data Types (Sybase IQ)

```
ENGINE SQLSYB SET VARCHAR {ON|OFF}
```

where:

SQLSYB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Maps the Sybase data types VARCHAR and VARBINARY as variable-length alphanumeric (A*n*V). ON is the default value.

OFF

Maps the Sybase data types VARCHAR and VARBINARY as alphanumeric (A).

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLSYB SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Reporting Against a Sybase Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Sybase stored procedures and report against a procedure's output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters, OUT parameters, and INOUT parameters in OUT mode, as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 2414.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 2418.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Note: For Sybase IQ data, support for stored procedures began with the SYBIQ 12.7 ESD #4 release.

Generating a Synonym for a Stored Procedure

A synonym describes a stored procedure's parameters and answer set.

An answer set's structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, "input parameters" refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception: if you know the internal logic of the procedure, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

Reference: Synonym Creation Parameters for Stored Procedures

Parameter/Task	Description
Restrict Object Type to	Select *Stored Procedures*.

Parameter/Task	Description
Database selection	To specify a database from which you can select a table or other object, do one of the following: ❑ Check *Use current database* to use the database that has been set as the default database. ❑ Select a database from the *Select database* drop-down list, which lists all databases in the current DBMS instance. Before selecting a database, if *Use current database* is checked, uncheck it. This option applies to Sybase ASE. It does not apply to Sybase IQ.
Filter by Owner/Schema and Object name	Selecting this option adds the Owner/Schema and Object Name parameters to the screen. **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end. **Object name.** Type a string for filtering the procedure names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all procedures whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.
Select	Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.
Name	The name of the synonym, which defaults to the stored procedure name.

Parameter/Task	Description
Select Application	Select an application directory. The default value is baseapp.
Prefix/Suffix	If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters. If all procedures have unique names, leave the prefix and suffix fields blank.
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.
Customize data type mappings	To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed. For information about them, see *Sybase Data Type Support* on page 2411.

Parameter/Task	Description
Values	Select the check box for *every* parameter displayed for the specified procedure. Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines: ❑ Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided. ❑ Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym-creation purposes. If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.

Creating a Report Against a Stored Procedure

You can report against a stored procedure answer set using the same facilities you use to report against a database table:

❑ **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 2420.

❑ **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 2419.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: **How to Report Against a Stored Procedure Using the TABLE Command**

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

IF

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

WHERE

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Sybase Environment

The Adapter for Sybase provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Activating NONBLOCK Mode

The Adapter for Sybase has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE SQLSYB SET NONBLOCK {0|n}
```

where:

`SQLSYB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLSYB SET PASSRECS {ON|OFF}
```

where:

`SQLSYB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Enabling Password Encryption for Sybase ASE and IQ

You can control password encryption when the adapter connects to a Sybase ASE or Sybase IQ data source. By default, the password encryption is turned off. If you activate the PASSWORD_ENCRYPTION setting, all subsequent connections made by the Adapter for Sybase servers are affected.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. When the Change Settings pane opens, choose OFF, ON, or EXTENDED from the pull-down list next to the PASSWORD_ENCRYPTION option.

Syntax: How to Enable Password Encryption (ASE and IQ)

```
SQL SQLSYB SET PASSWORD_ENCRYPTION {ON|EXTENDED|OFF}
```

where:

ON

Sets the connection property CS_SEC_ENCRYPTION. Use this setting when you are only configuring for connections to a Sybase ASE data source.

EXTENDED

Sets the connection property CS_SEC_EXTENDED_ENCRYPTION. Use this setting when you are only configuring for connections to a Sybase IQ data source or when you are configuring for connections to both Sybase ASE and Sybase IQ data sources.

- The property CS_SEC_EXTENDED_ENCRYPTION is available starting with Sybase Open Client version 15 ESD #7. If this value is not available for the given client version (that is, a client version earlier than Sybase Open Client version 15 ESD #7), the error message FOC1811 will be produced at the time of the setting and at the time of a connection attempt. (Other possible errors that may occur if the user's version of Open Client is earlier than 15 ESD #7 include the following: SQLCODE IS 5; ct_con_props(): User API layer: external error: An illegal value of 9213 given for parameter property.)
- Sybase IQ supports only CS_SEC_EXTENDED_ENCRYPTION. A Sybase error message will result if a connection to an IQ server is attempted with the value ON in effect: SQLCODE IS 4002; ASA Error -103: Invalid user ID or password.

OFF

Password encryption is disabled. OFF is the default setting.

Sybase Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Specifying Block Size for Retrieval Processing

The Adapter for Sybase supports array retrieval from result sets produced by executing SELECT queries or stored procedures. This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements. A value higher than 100 is not recommended because the increased efficiency it would provide is generally negligible.

Tip: You can change this setting manually or from the Web Console by clicking *Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the right-click menu. The Change Settings pane opens.

Syntax: How to Specify Block Size for Array Retrieval

The block size for a SELECT request applies to TABLE FILE requests, MODIFY requests, MATCH requests, and DIRECT SQL SELECT statements.

```
ENGINE SQLSYB SET FETCHSIZE n
```

where:

`SQLSYB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be retrieved at once using array retrieval techniques. Accepted values are 1 to 5000. The default varies by adapter. If the result set contains a column that has to be processed as a CLOB or a BLOB, the FETCHSIZE value used for that result set is 1.

Syntax: How to Specify Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE SQLSYB SET INSERTSIZE n
```

where:

`SQLSYB`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5000. 1 is the default value. If the result set contains a column that has to be processed as a BLOB, the INSERTSIZE value used for that result set is 1.

Syntax: How to Suppress the Bulk Insert API

```
ENGINE SQLSYB SET FASTLOAD [ON|OFF]
```

where:

SQLSYB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Uses the Bulk Insert API. ON is the default.

OFF

Suppresses the use of the Bulk Insert API.

Reference: Bulk Insert API Behavior

You can use DataMigrator with the Bulk Insert API for Sybase.

With the Adapter for Sybase, the Bulk Insert API is used in LOADONLY mode when INSERTSIZE is greater than 1. INSERTSIZE determines how often the intermediate data flush is performed. Measurements show that intermediate flushes are necessary for optimal Sybase server performance. As a rule of thumb, the INSERTSIZE value should be from several thousand to several tens of thousands. Intermediate flushes cannot be rolled back.

When you suppress the Bulk Load API, the Adapter for Sybase resorts to array insert according to the specified INSERTSIZE.

Errors that occur during the load (such as duplication) can cause the batch of rows to be rejected as a whole.

Calling a Sybase Stored Procedure Using SQL Passthru

Sybase stored procedures are supported using SQL Passthru. These procedures need to be developed within Sybase using the CREATE PROCEDURE command. A Sybase stored procedure is in either chained or unchained transaction mode. To change the default CHAINED property, see *How to Run an Unchained Sybase Stored Procedure* on page 2426.

Syntax: How to Call a Sybase Stored Procedure

The supported syntax to call a stored procedure is shown below. It is recommended that you use the syntax below instead of the previously supported CALLSYB syntax.

```
ENGINE SQLSYB
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE ON TABLE PCHOLD
END
```

where:

SQLSYB

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

Example: **Sybase Stored Procedure**

```
CREATE PROCEDURE SAMPLE
AS
SELECT SSN5, LAST_NAME5, FIRST_NAME5, BIRTHDATE5, SEX5 FROM EDAQA.NF29005
go
exec sp_procxmode 'PROC1','anymode'
go
```

Syntax: **How to Run an Unchained Sybase Stored Procedure**

Sybase ASE stored procedures may be in either chained or unchained transaction mode.

By default, the Adapters for Sybase run stored procedures in chained transaction mode. If you want to run an unchained Sybase stored procedure, you need to first set the adapter CHAINED property to OFF using the following command:

```
ENGINE SQLSYB SET CHAINED {OFF|ON};
END
```

where:

SQLSYB

Indicates the adapter ASE. You can omit this value if you had previously issued the SET SQLENGINE command.

OFF

Sets the adapter to unchained transaction mode.

ON

Sets the adapter to chained transaction mode.

The adapter is in chained transaction mode by default.

You can issue this command in a server procedure (focexec) or in the server profile (edasprof).

Chapter 100

Using the Adapter for Teradata

The Adapter for Teradata allows applications to access Teradata data sources. The adapter converts application requests into native Teradata statements and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Teradata Environment

To configure and use the ODBC or CLI Adapter for Teradata, the Teradata Client components must be installed and configured according to the *Teradata Tools and Utilities Installation Guide*. TeraGSS, ICU, and ODBC and/or CLI are required. The path to the ODBC or CLI libraries directory must be added to SYSTEM LIBRARY PATH prior to starting the server.

If you are using the z/OS server, the Teradata Channel-Attached CLI needs to be installed. Use the Web Console to configure the Teradata Adapter.

Procedure: How to Set Up the Environment on Microsoft Windows Using ODBC or CLI

On Microsoft Windows, the Teradata environment is set up during the installation of the product.

Procedure: How to Set Up the Environment on UNIX

You can access Teradata using the optional UNIX environment variable, ODBCINI. The ODBCINI variable points to the absolute path of the .odbc.ini file.

For example:

```
ODBCINI=/usr/odbc/.odbc.ini
export ODBCINI
```

Note: You must not use ODBCINI, if the $HOME directory contains the .odbc.ini file.

Configuring Teradata CLI and ODBC Wide API Adapters for Unicode

The Teradata CLI and ODBC Wide API adapters support Unicode UTF8 format if:

- The Teradata client components are part of release TTU12.0 or higher.
- The Teradata database is release 12.0 or higher and appropriate language support was enabled during the sysinit process.

Contact your DBA to determine whether international language support has been enabled in your Teradata system and/or consult the Teradata documentation for details about International Character Set Support.

Note that, at the present time, when Unicode is enabled the length of a Teradata Column Name and/or TITLE cannot exceed 21 characters.

Configuring the Adapter for Teradata

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Teradata database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Teradata database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to the Teradata Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Teradata

The *Teradata* adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Note that the release numbers on the Web Console for DBMS Add and DBMS Change panes refer to the Teradata ODBC driver, not the Teradata DBMS release.

Connection Name

This is the logical name used to identify this particular set of connection attributes.

Datasource or Server

Valid Teradata data source name (DSN). There is no default DSN; you must enter a value.

For the ODBC interface, this DSN name should match the User, System, or File DSN configured in the ODBC Administrator on Windows or the DSN entry in the $HOME/.odbc.ini file on UNIX.

For the CLI interface, this field is labeled SERVER and requires a valid Teradata TDP value. On Windows and UNIX, this is the value of i_dbcpath in the clispb.dat file. The default is *DBC* or <default>. This value may also be derived from the host file entry for the TD server. For example, if the host file contains td13cop1, then the server value will be td13. Refer to the *Teradata Tools and Utilities* documentation for details.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLDBC SET CONNECTION_ATTRIBUTES {connection DSN_name|Server}
/userid,password
```

where:

`SQLDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

This is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

For the Adapter for Teradata ODBC, this is the Teradata Data Source Name (DSN) you wish to access. It must match an entry in the odbc.ini file.

`Server`

For the Adapter for Teradata CLI, this is the Teradata Director Program number (TDP*n*), where *n* is the number. This is the value of i_dbcpath in the clispb.dat file.

`userid`

Is the primary authorization ID by which you are known to Teradata.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to the Teradata database server named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLDBC SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

Reference: Updating the Connection String for Teradata

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior 7.6.1 the connection was defined as:

  ```
  ENGINE SQLDBC SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

  ```
  ENGINE SQLDBC SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
  ```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLDBC SET DEFAULT_CONNECTION connection
```

where:

`SQLDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLDBC SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLDBC SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Teradata Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, create a synonym that describes the data source's structure and the server mapping of the Teradata data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each Teradata table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 2443.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Teradata

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Teradata Data Type Support* on page 2442.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Teradata synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLDBC ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=DSN_A,KEYS=1,WRITE=YES,$
```

Reference: Mapping Teradata Comments into a Synonym

When you generate a synonym for a Teradata table, the adapter maps each:

- Teradata table comment from the Teradata data dictionary to a Remark attribute in the Master File of the synonym.
- Teradata column comment from the Teradata data dictionary to a Description attribute in the Master File of the synonym.

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: TABLENAME=[*location.*][*owner.*]*tablename*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection*
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❏ KEYFLD is the FIELDNAME of the common column from the parent table. ❏ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.
AUTO INCREMENT	Set to Yes to enable auto-incrementing.
START	Initial value in the incrementing sequence.
INCREMENT	Increment interval.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: **Managing Synonyms**

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Teradata Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Controlling the Mapping of Variable-Length Data Types

The SET parameter VARCHAR controls the mapping of the Teradata data type VARCHAR. By default, the server maps this data type as variable character (A*n*V).

The following table lists data type mappings based on the value of VARCHAR:

		VARCHAR ON		VARCHAR OFF	
Teradata Data Type	**Remarks**				
VARCHAR (*n*)	*n* is an integer between 1 and 4000	A*n*V	A*n*V	A*n*	A*n*

Syntax: How to Control the Mapping of Variable-Length Data Types

```
ENGINE SQLDBC SET VARCHAR {ON|OFF}
```

where:

`SQLDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps the Teradata data type VARCHAR as variable-length alphanumeric (A*n*V). ON is the default value.

`OFF`

Maps the Teradata data type VARCHAR as alphanumeric (A).

Trailing Blanks in SQL Expressions

The new SQL Expression generator in the TABLE Adapter by default preserves literal contents, including trailing blanks in string literals and the fractional part and exponential notation in numeric literals. This allows greater control over the generated SQL.

In some rare cases when trailing blanks are not needed, the following syntax

```
ENGINE SQLDBC SET TRIM_LITERALS ON
```

is available to ensure backward compatibility.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Reporting Against a Teradata Stored Procedure

You can use a reporting tool, such as a SELECT statement or TABLE command, to execute Teradata stored procedures and report against a procedure output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- The retrieval of output parameters, (OUT and INOUT in OUT mode), as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

1. **Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure* on page 2443.
2. **Create a report procedure,** as described in *Creating a Report Against a Stored Procedure* on page 2446.
3. **Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

A synonym describes stored procedure parameters and answer set.

An answer set's structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, "input parameters" refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception: if you know the procedure's internal logic, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- INPUT, which describes any IN parameters and INOUT parameters in IN mode.

  If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.

- OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.

  If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.

- ANSWERSET*n*, one for each answer set.

  If there is no answer set, the segment is omitted.

Reference: Synonym Creation Parameters for Stored Procedures

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Select *Stored Procedures*.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.

- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Select

Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.

Name

The name of the synonym, which defaults to the stored procedure name.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed characters.

If all procedures have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Teradata Data Type Support* on page 2442.

Values

Select the check box for *every* parameter displayed for the specified procedure.

Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines:

- Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided.
- Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym-creation purposes.

If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.

Creating a Report Against a Stored Procedure

You can report against a stored procedure's answer set using the same facilities you use to report against a database table:

- **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT* on page 2448.
- **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command* on page 2446.

When joining from or to a stored procedure answer set, you can:

- **Join from** only OUTPUT and ANSWERSET segments in a host file.
- **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

IF

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

```
SQL
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
  .
  .
  .
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

WHERE

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode.

You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Teradata Environment

The Adapter for Teradata provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Controlling the Column Heading in a Request

You can specify a column's heading in a report using the Web Console. Alternatively, you can issue the SET COLNAME command.

Syntax: How to Control Column Headings Using a Command

```
ENGINE SQLDBC SET COLNAME {NAME|TITLE}
```

where:

`SQLDBC`

Indicates the adapter. You can omit this parameter value if you previously issued the SET SQLENGINE command.

`NAME`

If you specify the NAME option, the Teradata column name will be used as the column heading.

`TITLE`

If you specify the TITLE option, the name you provide is used as the column heading. TITLE is the default value.

Activating NONBLOCK Mode

The Adapter for Teradata has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Activate NONBLOCK Mode

```
ENGINE SQLDBC SET NONBLOCK {0|n}
```

where:

SQLDBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.
- Kill Session button on the Web Console is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLDBC SET PASSRECS {ON|OFF}
```

where:

SQLDBC

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Setting the Transaction Mode Within a Teradata Connection

You can control a Teradata connection's transaction mode using the Web Console or by issuing a command.

Syntax: How to Set the Transaction Mode of a Teradata Connection

To set the transaction mode of a Teradata connection, issue the following command in any profile or in a procedure

```
SQL SQLDBC SET TRANSACTION [ANSI|BTET|DEFAULT]
```

where:

`ANSI`

Sets the Teradata connection to the ANSI transaction mode. This is the default for a CLI connection.

`BTET`

Sets the Teradata connection to the Teradata transaction mode (also known as the BTET transaction mode).

`DEFAULT`

For ODBC connections only. Sets the Teradata connection to the Teradata system default transaction mode. This is the default for an ODBC connection.

For more information about how the Teradata system default is determined, see your Teradata documentation.

Note: The connection scope is different when setting the transaction mode for ODBC and CLI connections.

- **For an ODBC Connection:** The transaction mode can be set within an active connection. There is no needs to disconnect.
- **For a CLI Connection:** The transaction mode can be set only before the connection takes place. You must terminate the connection first using the SQL SQLDBC END SESSION command if you want to change the transaction mode for a CLI connection. Failure to do so will return the following warning:

```
(FOC1722) WARNING: COULD NOT SET TRANSACTION MODE
```

Teradata Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109 and *Optimizing Requests to Pass Virtual Fields Defined as Constants* on page 112.

Specifying the Block Size for Insert Processing

The Adapter for Teradata supports array insert if the release of Teradata Client is:

- **ODBC:** TTU13.10 or higher.
- **CLI:** TTU12.00 or higher.

This technique substantially reduces network traffic and CPU utilization.

Using high values increases the efficiency of requests involving many rows, at the cost of higher virtual storage requirements.

Tip: You can change this setting manually.

Syntax: How to Specify the Block Size for Insert Processing

In combination with LOADONLY, the block size for an INSERT applies to MODIFY INCLUDE requests. INSERTSIZE is also supported for parameterized DIRECT SQL INSERT statements.

```
ENGINE SQLDBC SET INSERTSIZE n
```

where:

`SQLDBC`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

n

Is the number of rows to be inserted using array insert techniques. Accepted values are 1 to 5,000. The default value is 1. If the result set contains a column that has to be processed as a LOB, the INSERTSIZE value used for that result set is 1.

Improving Efficiency With Aggregate Awareness

Aggregate awareness substantially improves the efficiency of queries.

Note: For Teradata, aggregate awareness is handled internally. No adapter settings are required. However, the COLLECT STATISTICS command must be issued in Teradata.

Optimizing Non-Equality WHERE-Based Left Outer Joins

A left outer join selects all records from the host table and matches them with records from the cross-referenced table. When no matching records exist, the host record is still retained, and default values (blank or zero) are assigned to the cross-referenced fields. The adapter can optimize any WHERE-based left outer join command in which the conditional expression is supported by the RDBMS.

Syntax: How to Specify a Conditional Left Outer JOIN

```
JOIN LEFT_OUTER FILE hostfile AT hfld1 [TAG tag1]
     [WITH hfld2]
     TO {UNIQUE|MULTIPLE}
     FILE crfile AT crfld [TAG tag2] [AS joinname]
     [WHERE expression1;
     [WHERE expression2;
     ...]

END
```

where:

`LEFT_OUTER`

Specifies a left outer join. If you do not specify the type of join in the JOIN command, the ALL parameter setting determines the type of join to perform.

`hostfile`

Is the host Master File.

`AT`

Links the correct parent segment or host to the correct child or cross-referenced segment. The field values used as the AT parameter are not used to cause the link. They are used as segment references.

`hfld1`

Is the field name in the host Master File whose segment will be joined to the cross-referenced data source. The field name must be at the lowest level segment in its data source that is referenced.

`tag1`

Is the optional tag name that is used as a unique qualifier for fields and aliases in the host data source.

`WITH hfld2`

Is a data source field with which to associate a DEFINE-based conditional JOIN. For a DEFINE-based conditional join, the KEEPDEFINES setting must be ON, and you must create the virtual fields before issuing the JOIN command.

`MULTIPLE`

Specifies a one-to-many relationship between *from_file* and *to_file*. Note that ALL is a synonym for MULTIPLE.

`UNIQUE`

Specifies a one-to-one relationship between *hostfile* and *crfile*. Note that ONE is a synonym for UNIQUE.

Note: Unique returns only one instance and, if there is no matching instance in the cross-referenced file, it supplies default values (blank for alphanumeric fields and zero for numeric fields).

The unique join is a WebFOCUS concept. The RDBMS makes no distinction between unique and non-unique situations; it always retrieves all matching rows from the cross-referenced file.

If the RDBMS processes a join that the request specifies as unique, and if there are, in fact, multiple corresponding rows in the cross-referenced file, the RDBMS returns all matching rows. If, instead, optimization is disabled so that WebFOCUS processes the join, a different report results because WebFOCUS, respecting the unique join concept, returns only one cross-referenced row for each host row.

crfile

Is the cross-referenced Master File.

crfld

Is the join field name in the cross-referenced Master File. It can be any field in the segment.

tag2

Is the optional tag name that is used as a unique qualifier for fields and aliases in the cross-referenced data source.

joinname

Is the name associated with the joined structure.

expression1, expression2

Are any expressions that are acceptable in a DEFINE FILE command. All fields used in the expressions must lie on a single path.

Reference: Conditions for WHERE-Based Outer Join Optimization

- In order for a WHERE-based left outer join to be optimized, the expressions must be optimizable for the RDBMS involved and at least one of the following conditions must be true:
  - The JOIN WHERE command contains at least one *field1* EQ *field2* predicate in which *field1* is in *table1* and *field2* is in *table2*.

    or
  - The right table has a key or a unique index that does not contain NULL data.

    or
  - The right table contains at least one "NOT NULL" column that does not have a long data type (such as TEXT or IMAGE).
- The adapter SQLJOIN OUTER setting must be ON (the default).

Example: Optimizing a Non-Equality Left-Outer Join

The following request creates a left outer conditional join between two Teradata data sources and reports against the joined data sources. The STMTRACE is turned on in order to view the SQL generated for this request:

```
SET TRACEUSER = ON
SET TRACEOFF = ALL
SET TRACEON = STMTRACE//CLIENT
JOIN LEFT_OUTER FILE baseapp/EQUIP AT CARS
TO ALL FILE baseapp/CARREC AT CARC
WHERE CARS NE CARC;
END
TABLE FILE baseapp/EQUIP
PRINT CARS CARC STANDARD
BY MODEL
END
```

The WebFOCUS request is translated to a single Teradata SELECT statement that incorporates the left outer join, and the non-equality condition is passed to the RDBMS in the ON clause:

```
SELECT T1."CARS"(CHAR( 16)),T1."STANDARD"(CHAR( 40)),
T2."CARC"(CHAR( 16)),T2."MODEL"(CHAR( 24)) FROM ( EQUIP T1 LEFT
OUTER JOIN CARREC T2 ON (T1."CARS" <> T2."CARC") ) ORDER BY
T2."MODEL";
```

Calling a Teradata Macro or Stored Procedure Using SQL Passthru

SQL Passthru is supported for Teradata macros and stored procedures.

Macros need to be developed within Teradata using the CREATE or REPLACE MACRO command. Procedures need to be developed within Teradata using the CREATE PROCEDURE command.

You must call a macro in the same transaction mode in which it was compiled. For information about setting the transaction mode, see *Setting the Transaction Mode Within a Teradata Connection* on page 2451. To find out which transaction mode is currently in effect, issue the HELP session command:

```
SQL SQLDBC HELP SESSION;
TABLE FILE SQLOUT PRINT TRANSACTION_SEMANTICS
END
```

Before you can call a stored procedure or a macro, you must set the connection accordingly, either using the Web Console or by issuing the SET MACRO command

```
SQL SQLDBC SET MACRO ON|OFF
```

where:

ON

Enables one to call a macro. This is the default.

OFF

Enables one to call a stored procedure.

Example: Calling a Macro

This is an example of the syntax for calling a macro:

```
ENGINE SQLDBC
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE FILE SQLOUT
END
```

Example: Calling a Stored Procedure

The supported syntax to call a stored procedure is shown below.

```
ENGINE SQLDBC
EX SAMPLE PARM1,PARM2,PARM3...;
TABLE FILE SQLOUT
END
```

When using the adapter with:

- **ODBC,** all scalar parameters (IN, OUT, and INOUT) are supported.
- **CLI,** scalar IN parameters, and INOUT parameters in IN mode, are supported.

Chapter 101

Using the Adapter for Transoft

The Adapter for Transoft allows applications to access Transoft data sources. The adapter converts application requests into Transoft calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the Transoft Environment

In order to use the Adapter for Transoft, you must install the Transoft driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the Transoft Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

   ```
   set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
   ```

   Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

   Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server's EDAPRINT. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for Transoft

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a Transoft data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one Transoft data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Transoft

The Transoft adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the Transoft data source.

Driver name

Name for the Transoft JDBC driver.

For example: jdbc.transoft.Driver

See Transoft documentation for the specific driver release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLTRN SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

SQLTRN

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection name.

URL

Is the URL to the location of the Transoft data source.

userid

Is the primary authorization ID by which you are known to the target database.

password

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the Transoft Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

Note: Consult vendor documentation for the exact name, port, and path.

```
ENGINE SQLTRN SET CONNECTION_ATTRIBUTES CON1
  'jdbc:transoft://hostname:7000/books.udd'
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLTRN SET DEFAULT_CONNECTION connection
```

where:

SQLTRN

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLTRN SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLTRN SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

`SQLTRN`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing Transoft Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLTRN to identify the Adapter for Transoft.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLTRN [,$]
```

where:

file

Is the file name for the Master File. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

`SQLTRN`

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using Transoft, you must locally install the RDBMS Transoft Driver.

The Server can access third-party database tables across the Transoft network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Note: If you are creating a synonym for a Transoft data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Transoft

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.

- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 2472.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for Transoft synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLTRN ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.

Keyword	Description
TABLENAME	Identifies the Transoft table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❏ KEYFLD is the FIELDNAME of the common column from the parent table. ❏ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the Transoft Environment

The Adapter for Transoft provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to Transoft.

Syntax: How to Issue the TIMEOUT Command

```
ENGINE SQLTRN SET TIMEOUT {nn|0}
```

where:

`SQLTRN`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: How to Cancel Long Requests

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLTRN SET PASSRECS {ON|OFF}
```

where:

`SQLTRN`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Transoft Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 102

Using the Adapter for Twitter

This section describes how to configure the Twitter Adapter.

In this chapter:

Twitter Adapter Overview

You can configure the Twitter Adapter using the WebFOCUS Reporting Server Web Console. The adapter requires a connection, which stores the access token. A valid Twitter access token is required to issue Twitter API calls. The token is associated with a Twitter application and a Twitter Screen Name.

Creating a Twitter Application

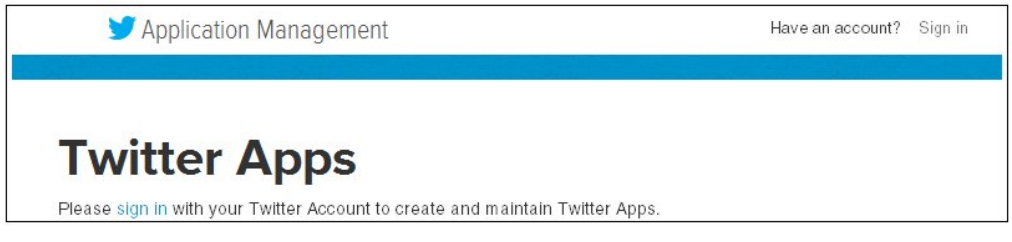
A Twitter application needs to exist before configuring the Twitter adapter

Procedure: How to Create a Twitter Application

1. From a browser, enter the following URL in a web browser:

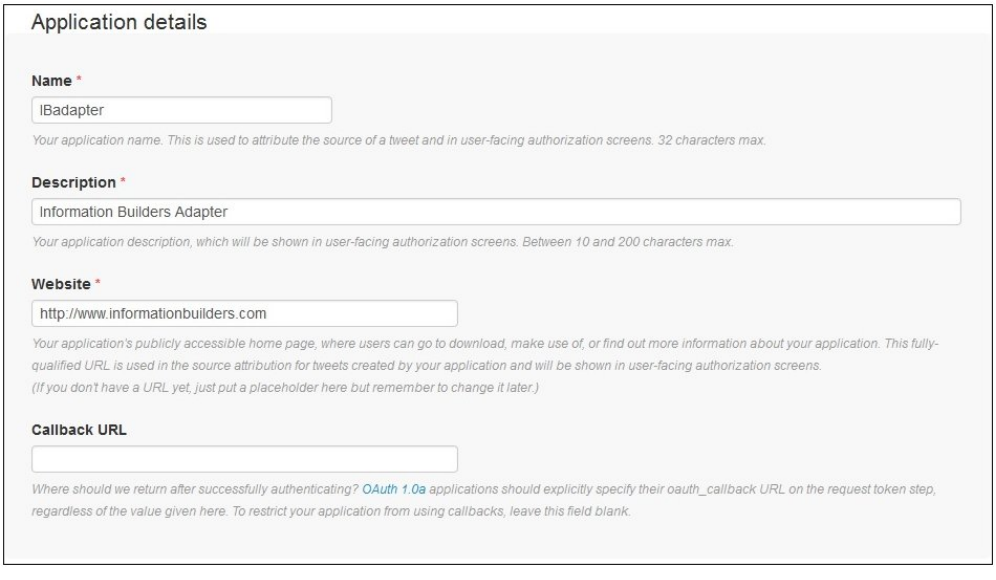
   `https://apps.twitter.com`

   A Twitter Application Management screen will appear if you are not already logged into Twitter, as shown in the following image.



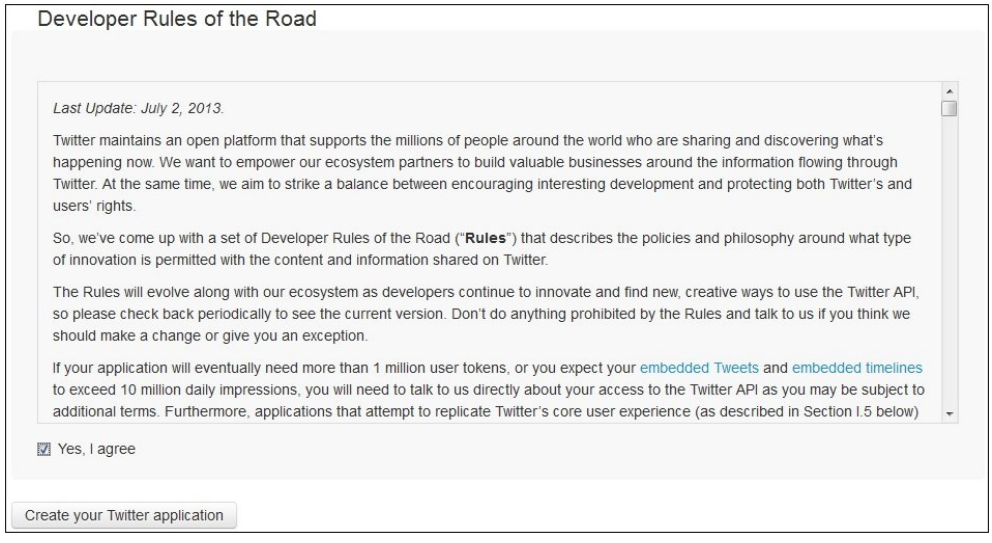
2. If you are not already signed in to Twitter, click the *Sign in* link located in the upper-right corner of the screen.
3. Click the *Create New App* button in the upper-right corner of the screen.

The Application details screen opens, as shown in the following image.



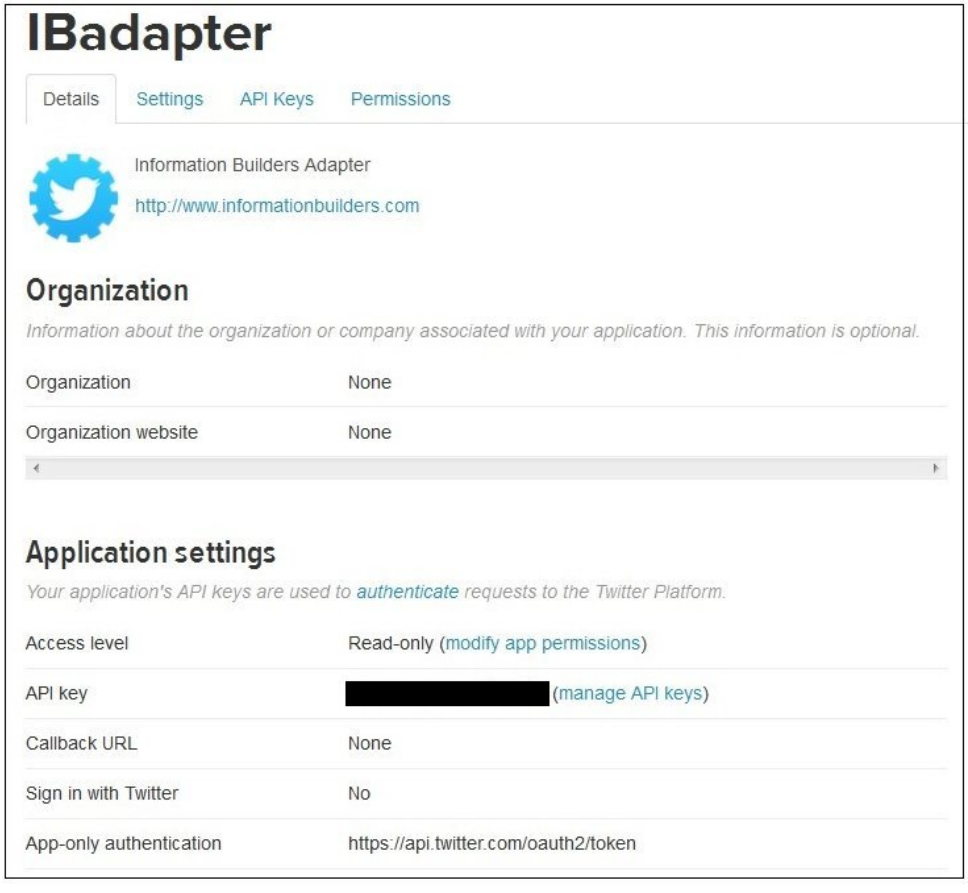
4. Perform the following steps:
   a. Enter a name without spaces for the new Twitter application in the Name field.
   b. Enter a description for the new Twitter application in the Description field.
   c. Enter web site URL for the company hosting the new Twitter application in the Website field.

5. Scroll down to the Developer Rules of the Road and read the agreement, as shown in the following image.



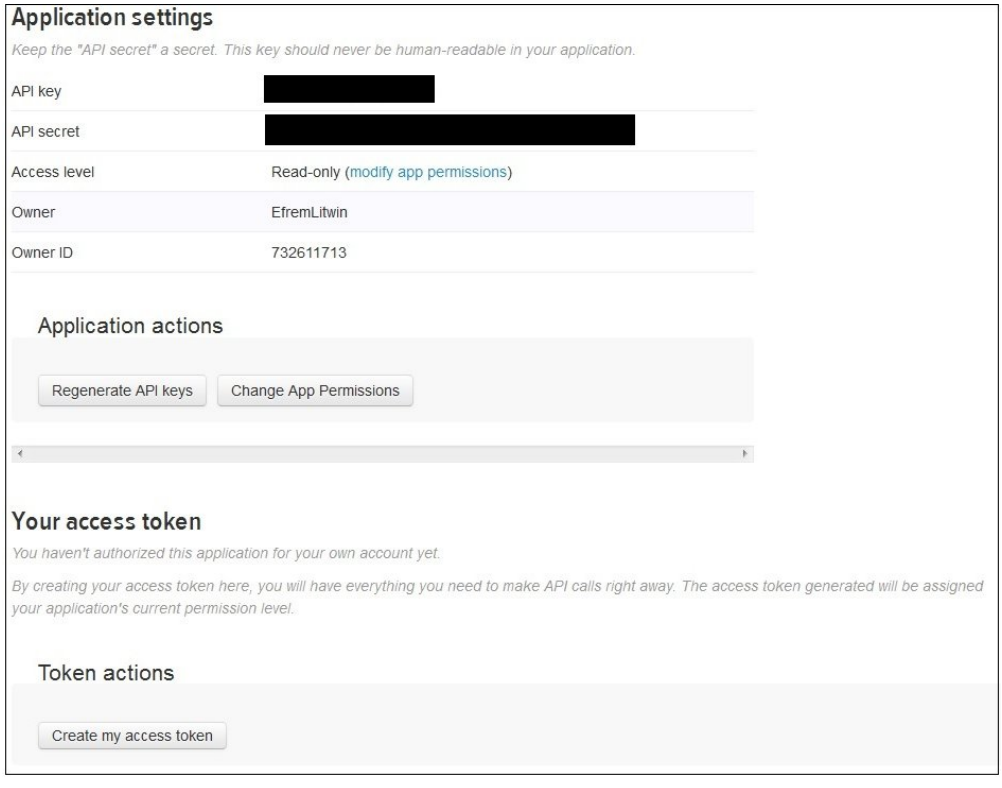
If the agreement is acceptable, select *Yes, I agree* and then click *Create your Twitter application*.

The configuration page for the new Twitter application (for example, IBadapter) opens, as shown in the following image.



6. Click the *manage API keys* link in the API key row.

The Application settings page opens, as shown in the following image.



Note: The *API key* and the *API secret* values will be required during the configuration of the Twitter adapter.

7. Click *Create my access token*.

   A Status message may appear at the top of the page, as shown in the following image.

8. Click the *Refresh* link within the status message.

9. Scroll down to the Your access token section of the page, as shown in the following image.

Note: The Access token and Access token secret values will be required during the configuration of the Twitter adapter.

Configuring the Twitter Adapter

This section describes how to configure the Twitter Adapter.

Procedure: How to Configure the Twitter Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

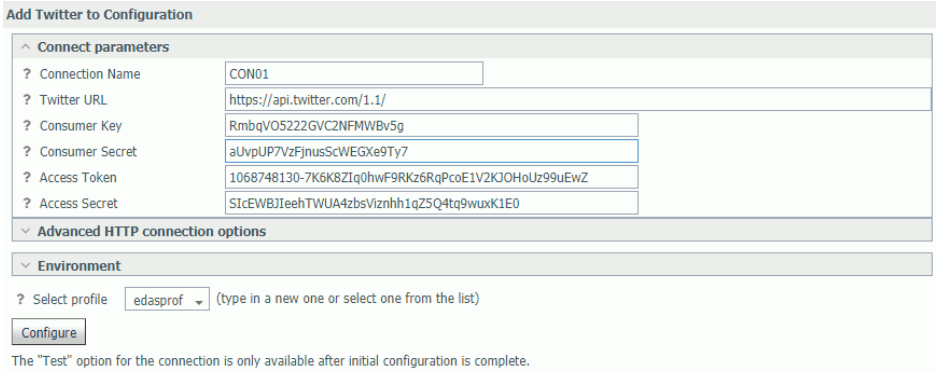
   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the *Twitter* node and select *Configure*.

The Add Twitter to Configuration pane opens, as shown in the following image.



5. Enter the values for the Consumer Key, Consumer Secret, Access Token, and Access Secret as defined in the Twitter application.

   For more information, see *Creating a Twitter Application* on page 2475.

6. Click *Configure*.

   The Twitter adapter is added to the configured Adapters list in the navigation pane.

Reference: Connection Attributes for Twitter

The following list describes the connection attributes for the Twitter adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Twitter URL

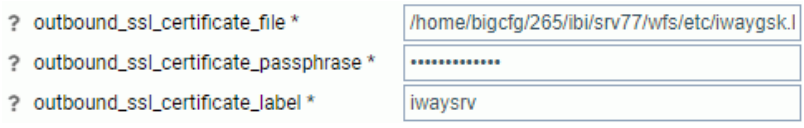
The URL of the Twitter API request. The default value is:

```
https://api.twitter.com/1.1/
```

For iSeries machines, the WebFOCUS Reporting Server must be configured for SSL as follows:

1. From the Web Console sidebar, click *Workspace*.
2. From the menu bar, click *Workspace Set*, and select *Miscellaneous Settings*

3. Enter values for *outbound_ssl_certificate_file*, *outbound_ssl_certificate_passphrase*, and *outbound_ssl_certificate_label*, and then click *Save*. For example:



Consumer Key

Also known as the API key, this application key is generated when creating a Twitter application. For more information, see *Creating a Twitter Application* on page 2475.

This key is used along with the Consumer Secret for authentication purposes.

Consumer Secret

Used for authentication purposes, along with the Consumer Key, this value is generated when creating a Twitter application. For more information, see *Creating a Twitter Application* on page 2475.

Access Token

Used for authentication purposes, along with the Access Secret, this key is generated when creating an Access Token from the Twitter application. For more information, see *Creating a Twitter Application* on page 2475. It defines the user who is authenticating to Twitter.

Access Secret

Used for authentication purposes, along with the Access Token, this value is generated when creating an Access Token from the Twitter application. For more information, see *Creating a Twitter Application* on page 2475.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the connection attributes. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), choose New Profile from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

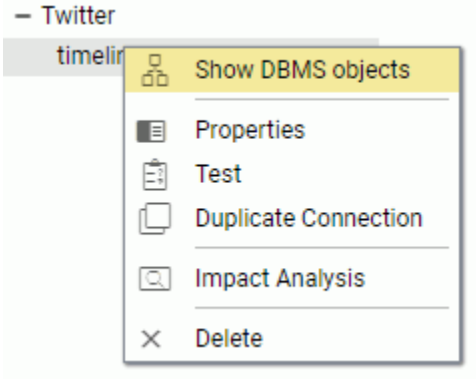
Store the connection attributes in the server profile (edasprof).

Creating Metadata and Sample Reports for the Twitter Adapter

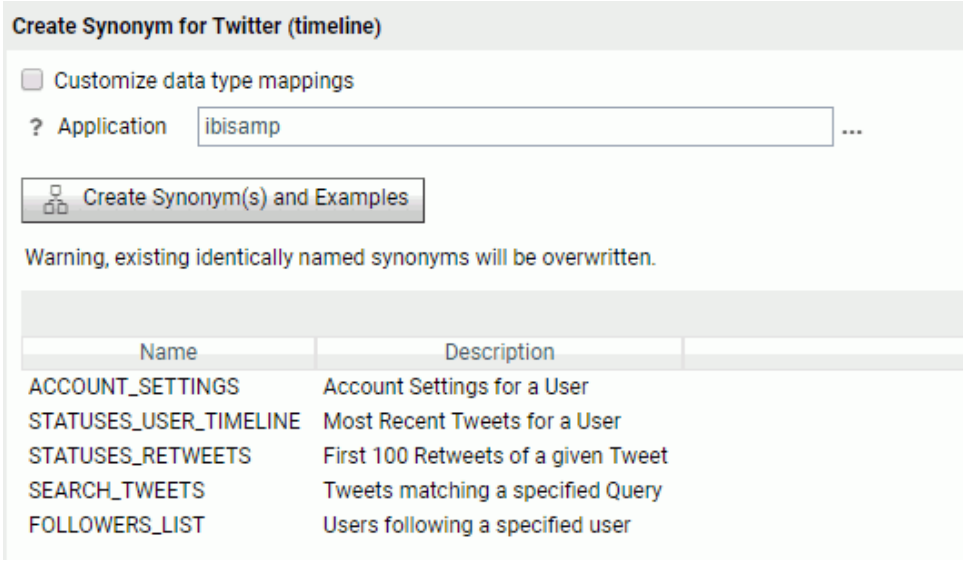
Create Synonym for the Twitter Adapter creates the metadata used for WebFOCUS reporting and DataMigrator ETL flows. It also creates sample WebFOCUS reports and DataMigrator ETL flows.

Procedure: How to Create Metadata and Sample Reports

1. From the WebFOCUS Reporting Server Web Console, expand the *Adapters* folder, *Configured* folder, and then the *Twitter* folder.
2. Right-click the configured connection for the Twitter Adapter (for example, twitter) and select *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for Twitter pane opens, as shown in the following image.



3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata, sample reports, and DataMigrator ETL flows are to be stored.
4. Click *Create Synonym(s) and Examples*.

   The Create Synonym for Twitter Status pane opens and indicates that the synonym was created successfully.

Twitter Examples

This section describes the metadata and sample reports for the Twitter Adapter.

Reference: Twitter Adapter Metadata

The following table lists and describes the available metadata for the Twitter Adapter.

Metadata	Description
account_settings	Used to retrieve the account settings for a Twitter user.
followers_list	Used to retrieve a list users following a specific user.

Metadata	Description
search_tweets	Used to search for tweets based on a query string.
statuses_retweets	Used to retrieve the first 100 retweets of a specific tweet.
statuses_user_timeline	Used to retrieve the most recent tweets for a Twitter Screen Name.
twtsampl/tw_tweet_sentiment	Describes a SQL Server table loaded by the tw_datamigrator_sentiment_load DataMigrator flow. The table contains Sentiment Scores for Twitter tweets. Used when the WAND Sentiment Analysis Adapter is configured.
twtsampl/tw_tweets	Describes a SQL Server table loaded by the tw_datamigrator_load DataMigrator flow. The table contains Twitter tweets.
twtsampl/tw_tweets_to_sentiment	Cluster Join from tw_tweets to wandscore; wandscore metadata is created from the WAND Sentiment Analysis Adapter. Used by the tw_datamigrator_sentiment_load DataMigrator flow.
twtsampl/tw_user_info	Describes a SQL Server table loaded by the tw_datamigrator_load DataMigrator flow. The table contains user information.

Reference: **Twitter Adapter Sample Reports**

The following table lists and describes the sample reports for the Twitter Adapter.

Sample Report	Description
twtsampl/account_settings	Retrieves the account settings for a Twitter user. Uses: account_settings

Sample Report	Description
twtsampl/followers_list	Displays a list of users who a follow a specific Twitter user. Uses: followers_list

Sample Report	Description

twtsampl/search_tweets	Search for all public posts containing a specified query string.
	Query operators:
	`word1 word2`
	Containing both words. This is the default operator.
	`"word1 word2"`
	Containing the exact phrase.
	`word1 OR word2`
	Containing either word1 or word2.
	`word1 -word2`
	Containing word1 but not word2.
	`#word1`
	Containing hashtag word1.
	`from:`*`ScreenName`*
	Tweets sent from a specific Screen Name.
	`to:`*`ScreenName`*
	Tweets sent to a specific Screen Name.
	`@ScreenName`
	Referencing a specific Screen Name.
	`word1 since:YYYY-MM-DD`
	Containing word1 and sent starting at a specific date.
	`word1 until:YYYY-MM-DD`
	Containing word1 and sent before a specific date.
	Uses search_tweets.

Sample Report	Description
twtsampl/statuses_retweets	Reports on the first 100 retweets of a specific tweet. Uses: statuses_retweets
twtsampl/statuses_user_timeline	Reports on the tweets for a specific Screen Name. Uses: statuses_user_timeline
twtsampl/tw_create_datamodel	Creates the SQL Server tables used for the Twitter Data Model loaded by the tw_datamigrator_load and tw_datamigrator_sentiment_load DataMigrator flows. As a prerequisite, a SQL Server connection called *socialmedia* must be configured as well as the creation of a SQL Server database called *Twitter*.
twtsampl/tw_delete_datamodel	Deletes the SQL Server tables used for the Data Model.
twtsampl/tw_tagcloud_tweets	Tag cloud graph for tweets. The Words Analysis adapter must be configured. Uses: twtsampl/tw_join_datamodel_excluding_sentiment wan_document (Words Analysis metadata)
twtsampl/tw_tweet_report	Reports on Twitter tweets and user information. Uses: twtsampl/tw_tweets twtsampl/tw_user_info

Sample Report	Description
twtsampl/tw_tweet_scored_report	Reports on Twitter tweets and user information including sentiment. Uses: twtsampl/tw_tweets twtsampl/tw_user_info twtsampl/tw_tweet_sentiment

Reference: **Twitter Adapter DataMigrator Flows**

The following table lists and describes the DataMigrator flows for the Twitter Adapter. The Twitter Data Model must first be created by running twtsampl/tw_create_datamodel.

Flow	Description
twtsampl/tw_datamigrator_load	Data flow to load Twitter tweets and user information.
twtsampl/ tw_datamigrator_sentiment_load	Data Flow to load Sentiment Scores for Twitter tweets.

Chapter 103

Using the Adapter for UniData

The Adapter for UniData allows applications to access UniData data sources. The adapter converts application requests into UniData calls and returns optimized answer sets to the requesting application.

In this chapter:

Preparing the UniData Environment

In order to use the Adapter for UniData, you must install the UniData driver, set its CLASSPATH value before server startup, and ensure that the JSCOM3 service is running.

Two additional steps are required to make UniData accessible to JDBC:

- All schema objects must be normalized by running CONVERT.SQL command in the native tool or via VSG(GUI-tool). This will make the SQLTables() call work properly.
- The old UniData 4.1 default schema must be migrated to current 6.0 and above schema-style using the MIGRATE.SQL command. This will make the SQLColumns() call work properly.

Procedure: How to Set Up the Environment on Windows and UNIX

1. Identify the location of the UniData Driver files by adding them to the environment variable CLASSPATH before server startup.

   For example, to set a UNIX or IBM i location for some JDBC Driver files to /usr/driver_files/mydriver.jar, issue the commands:

   ```
   CLASSPATH=/usr/driver_files/mydriver.jar;$CLASSPATH
   export CLASSPATH
   ```

   Similarly, on Windows, you would issue the command:

```
set CLASSPATH=c:\usr\driver_files\mydriver.jar:%CLASSPATH%
```

Note that if you run the server workspace as a Windows service, you must set the CLASSPATH as a system-wide environment variable.

Similarly, on UNIX, to ensure that the variable is set before server startup, add the CLASSPATH setting in your UNIX profile.

2. Start (or restart) the server.

   (Note that you also need to restart the server if the driver has changed.)

3. Verify that the JSCOM3 service is active by issuing the following command in the server workspace

   ```
   edastart -show
   ```

   and check the special services section displays "JSCOM3 active".

   If the JSCOM3 service is not active, a "fail to start" message will typically also be in the server's edaprint.log. To resolve the problem, review the JDBC listener requirements in the chapter for your operating system in the *Server Installation* manual.

Configuring the Adapter for UniData

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to a UniData data source, the adapter requires connection and authentication information.

You supply this information using the SET CONNECTION_ATTRIBUTES command. You can enter connection and authentication information in the Web Console or the Data Management Console configuration pane. The consoles add the command to the server profile you select: global (edasprof.prf), user (user.prf), or group (group.prf) if supported on your platform.

You can declare connections to more than one UniData data source using the SET CONNECTION_ATTRIBUTES commands. The actual connection takes place when the first query is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for UniData

The UniData adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

URL

Location URL for the UniData data source.

Driver name

Name for the UniData JDBC driver.

For example: com.ibm.u2.jdbc.UniJDBCDriver

See UniData documentation for the specific driver release you are using.

IBI_CLASSPATH

Defines the additional Java Class directories or full-path jar names which will be available for Java Services. Value may be set by editing the communications file or in the Web Console. Using the Web Console, you can enter one reference per line in the input field. When the file is saved, the entries are converted to a single string using colon (:) delimiters for all platforms. OpenVMS platforms must use UNIX-style conventions for values (for example, /mydisk/myhome/myclasses.jar, rather than mydisk:[myhome]myclasses.jar) when setting values. When editing the file manually, you must maintain the colon delimiter.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

Primary authorization ID by which you are known to the data source.

Password

Password associated with the primary authorization ID.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLUND SET CONNECTION_ATTRIBUTES connection
'URL'/userid,password
```

where:

`SQLUND`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection name.

`URL`

Is the URL to the location of the UniData data source.

`userid`

Is the primary authorization ID by which you are known to the target database.

`password`

Is the password associated with the primary authorization ID.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command connects to a data source using the UniData Driver, with an explicit user ID (MYUSER) and password (PASS). To ensure security, specifying connection attributes from the Web Console or DMC will encrypt the password before adding it to the server profile.

Note: Consult your vendor documentation for the exact name, port, and path.

```
ENGINE SQLUND SET CONNECTION_ATTRIBUTES CON1
  'jdbc:unidata://rediron6:7000/MYUSER,PASSSQLUND
```

Overriding the Default Connection

If multiple connections have been defined, the connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.

Syntax: How to Change the Default Connection

```
ENGINE SQLUND SET DEFAULT_CONNECTION connection
```

where:

SQLUND

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

connection

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLUND SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLUND SET AUTODISCONNECT ON {FIN|COMMIT}
```

where:

SQLUND

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMIT

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing UniData Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each object the server will access, you create a synonym that describes its structure and the server mapping of the data types.

Identifying the Adapter

The SUFFIX attribute in the Master File identifies the adapter needed to interpret a request. Use the SUFFIX value SQLUND to identify the Adapter for UniData.

Syntax: How to Identify the Adapter

```
FILE[NAME]=file, SUFFIX=SQLUND [,$]
```

where:

file

Is the file name for the Master File. The file name without the .mas extension can consist of a maximum of eight alphanumeric characters. The file name should start with a letter and be representative of the table or view contents. The actual file must have a .mas extension, but the value for this attribute should not include the extension.

SQLUND

Is the value for the adapter.

Accessing Database Tables

If you choose to access a remote third-party table using UniData, you must locally install the RDBMS UniData Driver.

The Server can access third-party database tables across the UniData network. You must specify a URL for the data source and, possibly, a user ID and/or password for the database you are accessing. You can define these parameters in either the server global profile or in a user profile.

Creating Synonyms

Synonyms define unique names (or aliases) for each object that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Note: If you are creating a synonym for a UniData data source, you must first add the syntax SET SYNONYM=BASIC to the edasprof.prf file.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for UniData

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

Build cluster using foreign keys (deprecated)

You can select the *Build cluster using foreign keys* check box to include within this synonym every table related to the current table by a foreign key. However, this option has been deprecated, as the recommended way to create a cluster is by using the Synonym Editor. The resulting multi-table synonym describes all of the foreign key relationships of this table.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *Data Type Support* on page 2504.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for UniData synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLUND ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION4, I9, I4, MISSING=OFF ,$
FIELD=DIVISION_HE4, DIVISION4, I9, I4, MISSING=OFF ,$
```

Access File nf29004.acx

```
SEGNAME=SEG1_4,TABLENAME=EDAQA.NF29004,
CONNECTION=CON1,KEYS=1,WRITE=YES,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.

Keyword	Description
TABLENAME	Identifies the UniData table. The value assigned to this attribute can include the name of the owner (also known as schema) and the database link name as follows: TABLENAME=[*owner.*]*table*
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local data source. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.

Keyword	Description
KEYFLD IXFLD	Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair. ❑ KEYFLD is the FIELDNAME of the common column from the parent table. ❑ IXFLD is the FIELDNAME of the common column from the related table. KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same. **Note:** An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Data Type Support

Data types are specific to the underlying data source.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the UniData Environment

The Adapter for UniData provides several parameters for customizing the environment and optimizing performance.

Specifying a Timeout Limit

TIMEOUT specifies the number of seconds the adapter will wait for a response after you issue an SQL request to UniData.

Syntax: **How to Issue the TIMEOUT Command**

```
ENGINE SQLUND SET TIMEOUT {nn|0}
```

where:

`SQLUND`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

nn

Is the number of seconds before a time-out occurs. 30 is the default value.

`0`

Represents an infinite period to wait for a response.

Cancelling Long Requests

You can cancel long running requests from the Web Console. Depending on the capabilities of the native JDBC driver, this action will either cancel the request entirely or break out of the fetch cycle.

Procedure: **How to Cancel Long Requests**

1. From the Web Console menu bar choose *Workspace*, *Configuration/Monitor*, *Java Services*, DEFAULT. Right-click on *DEFAULT* and select *Agents*.
2. In the Java Services Agents pane, highlight a row with the jscomid you wish to kill, right-click on it and select *Stop.*

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLUND SET PASSRECS {ON|OFF}
```

where:

`SQLUND`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

UniData Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 104

Using the Adapter for UniVerse

The Adapter for UniVerse allows applications to access UniVerse data sources. The adapter converts data or application requests into native UniVerse statements and returns optimized answer sets to the requesting program.

In this chapter:

Preparing the UniVerse Environment

For the Adapter for UniVerse, no environment variables need to be set prior to starting the server. However the UniVerse ODBC, UniCall Interface (UCI), and UniRPC need to be installed and configured according to the RocketSoftware *UniVerse ODBC Guide*. The UniRPC daemon should be running on both client and server machines.

Note: The adapter is relational and can only access tables and views, not UV files. The UV files can be converted to tables or views as described in *UV SQL Administration for DBA*s.

Configuring the Adapter for UniVerse

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

In order to connect to an UniVerse database server, the adapter requires connection and authentication information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one UniVerse database server by including multiple SET CONNECTION_ATTRIBUTES commands. The actual connection to UniVerse Server takes place when the first query that references the connection is issued. If you issue multiple SET CONNECTION_ATTRIBUTES commands:

- The connection named in the *first* SET CONNECTION_ATTRIBUTES command serves as the default connection.
- If more than one SET CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* SET CONNECTION_ATTRIBUTES command.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.
2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for UniVerse

The UniVerse adapter is under the *SQL* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

Datasource

The data source name (DSN). There is no default data source name. You must enter a value.

Security

There are two methods by which a user can be authenticated when connecting to a database server:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to the database, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to the database, at connection time, for authentication.

User

User name by which you are known to UniVerse. It is ignored when connecting to a local UniVerse server.

Password

Password associated with the user name. It is ignored when connecting to a local UniVerse server.

Schema

UniVerse schema name including the entire path to it enclosed in single quotation marks.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

Syntax: How to Declare Connection Attributes Manually

```
ENGINE SQLUV SET CONNECTION_ATTRIBUTES connection
DSN_name/userid,password [;dbname]
```

where:

`SQLUV`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the logical name used to identify this particular set of connection attributes.

Note that one blank space is required between *connection* and DSN_*name*.

`DSN_`*`name`*

Is the UniVerse Data Source Name (DSN) you wish to access. It must match an entry in the uci.config or uvodbc.config file, depending on the UniVerse release. Please consult the corresponding version of the RocketSoftware *UniVerse ODBC Guide*.

`userid`

Is the primary authorization ID by which you are known to UniVerse.

`password`

Is the password associated with the primary authorization ID.

`dbname`

Also referred to as schema, is the name of the UniVerse database used for this connection. The database name, including path, must be inclosed in single quotation marks.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command declares connection CON1 to the UniVerse DSN named SAMPLESERVER with an explicit user ID (MYUSER) and password (PASS). To ensure security, specify connection attributes from the Web Console, which encrypts the password before adding it to the server profile.

```
ENGINE SQLUV SET CONNECTION_ATTRIBUTES CON1 SAMPLESERVER/MYUSER,PASS
```

Reference: Updating the Connection String

The syntax for the CONNECTION_ATTRIBUTES command for this adapter has been enhanced to include a logical connection name that is designed to support the porting of applications from development to production environments. This enhanced syntax may necessitate the migration of existing CONNECTION_ATTRIBUTES commands.

The Web Console Migrate option migrates your server settings to the newer release. To access this option, select *Workspace*, then *Configuration/Monitor* from the menu bar. Right-click *Migrate* from the *Server* folder in the navigation pane, and select *Configure*. On the Migrate pane, type the full path of the configuration instance directory (EDACONF) and click the *Migrate* button. This is the recommended approach.

If you choose *not* to use the Migrate option, please note the following information:

- Connection names declared prior to Version 7 Release 6.1 are supported.
- If you create a new connection for the purpose of creating new synonyms, your existing connections are re-saved in a new format, and the existing synonyms continue to work without any changes.
- If you add a new connection for the purpose of using an existing synonym, you must change the default logical connection name to match the value that is stored in the existing Access File attribute CONNECTION=*value*.

  For example, suppose that prior to Version 7 Release 6.1, the connection was defined as:

  ```
  ENGINE SQLUV SET CONNECTION_ATTRIBUTES DSN_A/uid,pwd
  ```

  When synonyms based on objects stored in this DSN_A are created, the Access Files contains the following description:

  ```
  CONNECTION=DSN_A
  ```

  If you then add a new connection, you must change the connection name from the default CON01 to DSN_A and save it as DSN_A in order to reuse the existing synonym. The connection is stored in the profile as:

```
ENGINE SQLUV SET CONNECTION_ATTRIBUTES DSN_A DSN_A/uid,pwd
```

Overriding the Default Connection

Once connections have been defined, the connection named in the first SET CONNECTION_ATTRIBUTES command serves as the default connection. You can override this default using the SET DEFAULT_CONNECTION command.

Syntax: How to Change the Default Connection

```
ENGINE SQLUV SET DEFAULT_CONNECTION connection
```

where:

`SQLUV`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`connection`

Is the connection defined in a previously issued SET CONNECTION_ATTRIBUTES command. If this name was not previously declared, the following message is issued:

```
FOC1671, Command out of sequence
```

Note:

- If you use the SET DEFAULT_CONNECTION command more than once, the connection name specified in the *last* command serves as the default connection.
- The SET DEFAULT_CONNECTION command cannot be issued while an uncommitted transaction (LUW) is pending. In that case, the following message is issued:

```
FOC1671, Command out of sequence.
```

Example: Selecting the Default Connection

The following SET DEFAULT_CONNECTION command selects the database server named SAMPLE as the default database server:

```
ENGINE SQLUV SET DEFAULT_CONNECTION SAMPLE
```

Controlling the Connection Scope

The SET AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

```
ENGINE SQLUV SET AUTODISCONNECT ON {FIN|COMMAND|COMMIT}
```

where:

`SQLUV`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`FIN`

Disconnects automatically only after the session has been terminated. FIN is the default value.

`COMMAND`

Disconnects automatically after each request. Depending on how often the event occurs, the SET AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server. It is related to the operating system and the data source.

`COMMIT`

Disconnects automatically only after COMMIT or ROLLBACK is issued as a native SQL command.

Managing UniVerse Metadata

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the data source's structure and the server mapping of the UniVerse data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each UniVerse table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for UniVerse

The following list describes the synonym creation parameters for which you can supply values.

Restrict Object Type to

Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.

Choosing *External SQL Scripts* from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments.

Depending on the adapter, you can further restrict your search by choosing check boxes for listed objects.

Filter by Owner/Schema and Object name

Selecting this option adds the Owner/Schema and Object Name parameters to the screen.

- **Owner/Schema.** Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.
- **Object name.** Type a string for filtering the object names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all objects whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.

Location of External SQL Scripts

If you specify *External SQL Scripts* in the *Restrict Object type to* field, these additional fields are displayed.

The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:

- In the *Base Location* field, specify the physical directory location of the file that contains the SQL Query. You can type a directory name or click on the ellipsis. This opens the Select Base Location dialog box.
- In the *Document Name* field, enter the file name with or without wild card characters.
- In the *Document Extension* field, enter the extension of the script files to filter the list of candidates.

On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:

- In the *Base Location* field, enter:

  ```
  /QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE
  ```

- The *Document Extension* is understood to be MBR. You can enter this value explicitly or leave the input box blank.

During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,

```
DATASET=/ul/home2/apps/report3.sql
```

When a WebFOCUS report is created, the SQL Query is used to access data.

Cardinality

Select the *Cardinality* check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.

If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.

For Subquery

Only available when *External SQL Scripts* is selected from the Restrict objects type to drop-down menu. When selected, a SUBQUERY keyword is added to the Access File of the generated synonym. If the corresponding SQL string has valid syntax that can be used in the FROM statement of the generated SQL (what is known as a Derived Table), then the SQL SCRIPT will be processed as a subquery embedded into a FROM clause. This usage allows for more flexibility. For example, the synonym can be used as a target for a JOIN.

If the SQL SCRIPT has parameter markers, such as ? or :, or the syntax contains constructs that are invalid for a derived table, for example ORDER BY, then this keyword should not be selected. At runtime, if SUBQUERY=Y is present and it is determined that the SQL SCRIPT cannot be used in the FROM statement, the setting will be ignored, and a FOC1782 warning message will be issued. The default is selected (SUBQUERY=Y).

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Customize data type mappings

To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.

For information about customizable mappings, see *UniVerse Data Type Support* on page 2519.

Update or Create Metadata

Select *Create* to overwrite any existing synonym with the same fully-qualified name, or *Update* to synchronize the metadata with an existing synonym. If you select *Update*, the next screen will show a list of attributes from the DBMS catalog that you can check to allow attributes from the DBMS catalog to override attributes from the existing synonym.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

Owner/Schema

The user account that created the object or a collection of objects owned by a user.

Table name

Is the name of the underlying object.

Type

The object type (Table, View, and so on).

Select tables

Select tables for which you wish to create synonyms:

- To select all tables in the list, select the *Select All* check box.
- To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

An Adapter for UniVerse synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLUV ,$
SEGNAME=SEG1_4, SEGTYPE=S0 ,$
FIELD=DIVISION4,    DIVISION4,    I9,  I4,  MISSING=OFF ,$
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON  ,$
FIELD=DIVISION_HE4, DIVISION_HE4, I9,  I4,  MISSING=ON  ,$
```

Generated Access File nf29004.acx

```
SEGNAME=NF29004 ,
TABLENAME=NF29004,
CONNECTION=CONN_UV,
KEYS=0
,$
```

Reference: Access File Keywords

This chart describes keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	Name of the table or view. This value can include a location or owner name as follows: TABLENAME=[location.][owner.]tablename

Keyword	Description
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection* CONNECTION=' ' indicates access to the local UniVerse database server. Absence of the CONNECTION attribute indicates access to the default database server.
KEYS	Indicates how many columns constitute the primary key for the table. Corresponds to the first *n* fields in the Master File segment. See the KEY attribute below for information about specifying the key fields without having to describe them first in the Master File.
KEY	Specifies the columns that participate in the primary key without having to describe them as the first fields in the Master File. The syntax is: KEY=*fld1/fld2/.../fldn*
WRITE	Specifies whether write operations are allowed against the table.
INDEX_NAME INDEX_UNIQUE INDEX_COLUMNS INDEX_ORDER	Indicate a name of the index in a database, uniqueness, name, and order of the indexed column(s).

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

UniVerse Data Type Support

SQL Data Type mapping options are available in a report available from the Web Console.

For more information, see *How to Access the Data Type Report* on page 95.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Customizing the UniVerse Environment

The Adapter for UniVerse provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Displaying Multivalued Columns in UniVerse

The OPENMODE NNF setting determines how tables with multivalued columns are treated. For the Adapter for Universe to function properly, OPENMODE must be set to NNF in edasprof.prf.

Syntax: How to Set OPENMODE NNF

```
ENGINE SQLUV SET OPENMODE NNF
```

where:

`SQLUV`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`OPENMODE NNF`

Allows the user application to see UniVerse tables the way they actually exist, including any multi-valued columns.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

Syntax: How to Obtain the Number of Rows Updated or Deleted

```
ENGINE SQLUV SET PASSRECS {ON|OFF}
```

where:

SQLUV

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

Controlling Transactions

In order to perform Data Definition Language commands in UniVerse using the server, the TRANSACTION setting must be set to OFF.

Syntax: How to Issue the TRANSACTIONS Command

```
ENGINE SQLUV SET TRANSACTIONS {ON|OFF}
```

where:

SQLUV

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

ON

Turns transaction processing on. ON is the default value.

OFF

Turns transaction processing off.

UniVerse Optimization Settings

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

For more information, see *Optimizing Requests* on page 109.

Chapter 105

Using the Adapter for VSAM

The Adapter for VSAM allows applications to access VSAM data sources. The adapter converts application requests into native VSAM statements and returns optimized answer sets to the requesting application or it inserts the data from an application to the data source. VSAM ESDS, KSDS, and RRDS data sources are supported in Read/Write mode.

The Adapter for VSAM can access VSAM files in batch mode (where VSAM clusters are allocated to the server).

You can also access compressed datasets with this adapter. However, the actual compression is done using the ZCOMP Exit. For details, see *Data Set Compression Exit: ZCOMP* on page 2719.

In this chapter:

Preparing the Environment for VSAM

The Adapter for VSAM does not require setting any environment variables.

You can configure the Adapter for VSAM from the Web Console without completing any preparation steps.

Configuring the Adapter for VSAM

You can configure the adapters from the Web Console or the Data Management Console.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.

6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Managing VSAM Metadata

When the server accesses a data source, it interprets the data stored there based on instructions in a synonym that you create, which describes the structure of the data source and the server mapping of the VSAM data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each VSAM file or view that is accessible from the server. Synonyms are useful because they hide the underlying data source location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File that represents the server metadata.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
   - **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.
3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for VSAM

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Select File System Option

Under File System Selection, choose one of the following options:

- *Fully qualified PDS name* to indicate a partitioned dataset on MVS.

  In the input boxes provided, enter a PDS name preceded by a double slash (//) and a Member name containing the location of the COBOL FD source. If you wish, you can filter the member name using a wildcard character (%).

  or

- *Absolute HFS directory pathname* to indicate a hierarchical file structure on USS.

  In the input boxes provided, type a Directory name to specify the HFS location that contains the COBOL FD and File name and File extension. If you wish, you can filter the file and extension using a wildcard character (%).

Click Submit

The Create Synonym pane (Step 2 of 2) opens.

If selected, the filtered COBOL definition is displayed at the top of the pane.

Enter the following parameters, as required:

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Set COBOL FD translation options

Optionally, select *Customize COBOL FD conversion options* to customize how the COBOL FD is translated. If you do not select the check box, default translation settings are applied.

For more information, see *Customization Options for COBOL File Descriptions* on page 2700.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Extended data format attributes

This option provides data portability between servers by enabling processing of data loaded in different encoding systems.

Click the *Extended data format attributes* check box to display the *Codepage* option.

In the input box provided, enter the code page of the stored data. Your entry is added to the Master File of the generated synonym.

Note: The code page you specify here must one for which you have requested a customized code page conversion table as part of your NLS configuration. If you have not already requested a conversion table for the designated code page, you can do so now. On the Web Console menu bar, click *Workspace*, then *Configuration*, and choose *NLS* under General settings. The NLS Configuration Wizard opens. Click *Customize code page conversion tables* and select the check box for the code page you wish to use. Click the *Save and Restart* button to implement your new setting.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Create Synonym parameters

Default Synonym Name

This column displays the names that will be assigned to each synonym. To assign different names, replace the respective displayed values. You have these additional options:

- To select all synonyms in the list, select the check box to the left of the Default Synonym Name column heading.
- Enter a cluster name under *Dataset Location* to associate it with a particular metadata description. The cluster name is embedded in the Master File. If you do not enter the cluster name during the synonym creation process, you will have to add it dynamically at run time.
- To select specific synonyms, select the corresponding check boxes.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Associating a VSAM Data Source With a Master File

You can associate a VSAM file with a Master File using the following methods:

- **DATASET attribute in a Master File.** The DATASET attribute in a Master File specifies the physical location for the data source to be allocated. This attribute is used at the file declaration level of the Master File. The CREATE SYNONYM facility uses the DATASET attribute in the Master File.
- **FILEDEF command.** An explicit allocation (FILEDEF) of data for the Master File is checked for when a DATASET attribute is detected. If an explicit allocation is found, the DATASET attribute is ignored and the allocation is used.

If the data file name is not allocated, an internal command is issued to perform the allocation using the DATASET value. This allocation is stored temporarily and is released when a new Master File is used or when the session terminates.

Syntax: How to Specify a Physical File Location for the VSAM Data Source

The DATASET attribute is used at the file declaration level of the Master File. The syntax is

```
{DATASET|DATA}='filename'
```

where:

filename

Is the platform-dependent physical name of the data source.

Note: If a DATASET allocation is in effect, a CHECK FILE command must be issued in order to override it by an explicit allocation command. The CHECK FILE command will deallocate the allocation created by DATASET.

Example: Allocating a VSAM Data Source Using the DATASET Attribute

The following example illustrates how to allocate a VSAM data source on the file declaration level and for an alternate index:

```
FILE=EXERVSM1, SUFFIX=VSAM, DATASET='VSAM1.CLUSTER1',$
SEGNAME=ROOT , SEGTYPE=S0,$
 GROUP=KEY1  , ALIAS=KEY , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD1 , ALIAS=F1  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD2 , ALIAS=F2  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD3 , ALIAS=DD1 , FORMAT=A4, ACTUAL=A4 , FIELDTYPE = I ,
        DATASET='VSAM1.INDEX1' ,$
  FIELD=FLD4 , ALIAS=F4  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD5 , ALIAS=F5  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD6 , ALIAS=F6  , FORMAT=A4, ACTUAL=A4 ,$
  FIELD=FLD7 , ALIAS=F7  , FORMAT=A4, ACTUAL=A4 ,$
```

Example: Overriding DATASET With a FILEDEF Command

You can override the DATASET value in the Master File with a FILEDEF command. The FILEDEF can be done locally within a procedure (before accessing the file) or within the profile. For example,

```
FILEDEF vsamfile DISK filename
```

where:

vsamfile

Is the logical metadata reference name matching the physical file name.

filename

Is the OpenVMS file name. It may be any valid OpenVMS file specification or logical name pointing to a file.

Syntax: How to Specify the Location of VSAM Files Manually

VSAM data sets may be allocated with JCL

```
//QAVSM DD DISP=SHR, DSN=cluster_name
```

or with DYNAM

```
DYNAM ALLOC FILE QAVSM DA cluster_name SHR REUSE
```

or in a Master File as

```
DATASET='dataset_name,' $
```

where:

cluster_name

Is the name of the VSAM cluster.

dataset_name

Is the name of the data set on MVS.

Note that if you are using an alternate index (an additional, separate index structure that enables you to access records in a VSAM data source based on a key other than the primary key for that data source), the alternate index must be associated with the base cluster it describes by a path that is stored in a separate data source, see *Using VSAM Relative Record Data Set (RRDS) Files* on page 2578. In addition to defining the location of the base cluster, you must define the path clusters either in IRUNJCL or using a DYNAM command. For details about logical processing with alternate indexes, see *Reporting From Files With Alternate Indexes* on page 2568.

Note: The Adapter for VSAM supports full read/write access.

Example: **Defining Path Clusters in JCL**

```
//VSAMFL1  DD DISP=OLD,DSN=EDABXV.VSAM.AXINDEX1.PATH
//VSAMFL2  DD DISP=OLD,DSN=EDABXV.VSAM.AXINDEX2.PATH
```

Standard Master File Attributes for a VSAM Data Source

Most standard Master File attributes are used with VSAM data sources in the standard way.

- **SUFFIX.** The SUFFIX attribute in the file declaration for these data sources has the value VSAM.
- **SEGNAME.** The SEGNAME attribute of the first or root segment in a Master File for a VSAM data source must be ROOT. The remaining segments can have any valid segment name.

  The only exception involves unrelated RECTYPEs, where the root SEGNAME must be DUMMY.

  All non-repeating data goes in the root segment. The remaining segments may have any valid name from 1 to 8 characters.

  Any segment except the root is the descendant, or child, of another segment. The PARENT attribute supplies the name of the segment that is the hierarchical parent or owner of the current segment. If no PARENT attribute appears, the default is the immediately preceding segment. The PARENT name may be one to eight characters.
- **SEGTYPE.** The SEGTYPE attribute should be S0 for VSAM data sources.
- **GROUP.** The keys of a VSAM data source are defined in the segment declarations as GROUPs consisting of one or more fields.

Describing a Group Field

A single-segment data source may have only one key field, but it must still be described with a GROUP declaration. The group must have ALIAS=KEY.

Groups can also be assigned simply to provide convenient reference names for groups of fields. Suppose that you have a series of three fields for an employee: last name; first name; and middle initial. You use these three fields consistently to identify the employee. You can identify the three fields in your Master File as a GROUP named EMPINFO. Then, you can refer to these three linked fields as a single unit, called EMPINFO. When using the GROUP feature for non-keys, the parameter ALIAS= must still be used, but should not equal KEY.

For group fields, you must supply both the USAGE and ACTUAL formats in alphanumeric format. The length must be exactly the sum of the subordinate field lengths.

The GROUP declaration USAGE attribute specifies how many positions to use to describe the key in a VSAM KSDS data source. If a Master File does not completely describe the full key at least once, the following warning message appears:

```
(FOC1016) INVALID KEY DESCRIPTION IN MASTER FILE
```

The cluster key definition is compared to the Master File for length and displacement.

When you expand on the key in a RECTYPE data source, describe the key length in full on the last non-OCCURS segment on each data path.

Do not describe a group with ALIAS=KEY for OCCURS segments.

If the fields that make up a group key are not alphanumeric fields, the format of the group key is still alphanumeric, but its length is determined differently. The ACTUAL length is still the sum of the subordinate field lengths. The USAGE format, however, is the sum of the internal storage lengths of the subordinate fields. You determine these internal storage lengths as follows:

- Fields of type I have a value of 4.
- Fields of type F have a value of 4.
- Fields of type P that are 8 bytes can have a USAGE of P15 or P16 (sign and decimal for a total of 15 digits). Fields that are 16 bytes have a USAGE of P17 or larger.
- Fields of type D have a value of 8.
- Alphanumeric fields have a value equal to the number of characters they contain as their field length.

Note:

- Since all group fields must be defined in alphanumeric format, those that include numeric component fields should not be used as verb objects in a report request.
- The MISSING attribute is not supported on the group field, but is supported on the individual fields comprising the group.

Syntax: How to Describe a VSAM Group Field

```
GROUP = keyname, ALIAS = KEY, USAGE = Ann, ACTUAL = Ann ,$
```

where:

keyname

Can have up to 66 characters.

Example: Describing a VSAM Group Field

In the library data source, the first field, PUBNO, can be described as a group key. The publisher's number consists of three elements: a number that identifies the publisher, one that identifies the author, and one that identifies the title. They can be described as a group key, consisting of a separate field for each element if the data source is a VSAM data structure. The Master File looks as follows:

```
FILE = LIBRARY5, SUFFIX = VSAM,$
 SEGMENT = ROOT, SEGTYPE = S0,$
  GROUP = BOOKKEY      ,ALIAS = KEY ,USAGE = A10   ,ACTUAL = A10  ,$
   FIELDNAME = PUBNO   ,ALIAS = PN  ,USAGE = A3    ,ACTUAL = A3   ,$
   FIELDNAME = AUTHNO  ,ALIAS = AN  ,USAGE = A3    ,ACTUAL = A3   ,$
   FIELDNAME = TITLNO  ,ALIAS = TN  ,USAGE = A4    ,ACTUAL = A4   ,$
  FIELDNAME = AUTHOR   ,ALIAS = AT  ,USAGE = A25   ,ACTUAL = A25  ,$
  FIELDNAME = TITLE    ,ALIAS = TL  ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = BINDING  ,ALIAS = BI  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PRICE    ,ALIAS = PR  ,USAGE = D8.2N ,ACTUAL = D8   ,$
  FIELDNAME = SERIAL   ,ALIAS = SN  ,USAGE = A15   ,ACTUAL = A15  ,$
  FIELDNAME = SYNOPSIS ,ALIAS = SY  ,USAGE = A150  ,ACTUAL = A150 ,$
  FIELDNAME = RECTYPE  ,ALIAS = B   ,USAGE = A1    ,ACTUAL = A1   ,$
```

Example: Describing a VSAM Group Field With Multiple Formats

```
GROUP = A, ALIAS = KEY, USAGE = A14, ACTUAL = A8   ,$
 FIELDNAME = F1, ALIAS = F1, USAGE = P6, ACTUAL=P2 ,$
 FIELDNAME = F2, ALIAS = F2, USAGE = I9, ACTUAL=I4 ,$
 FIELDNAME = F3, ALIAS = F3, USAGE = A2, ACTUAL=A2 ,$
```

The lengths of the ACTUAL attributes for subordinate fields F1, F2, and F3 total 8, which is the length of the ACTUAL attribute of the group key. The display lengths of the USAGE attributes for the subordinate fields total 17. However, the length of the group key USAGE attribute is found by adding their internal storage lengths as specified by their field types: 8 for USAGE=P6, 4 for USAGE=I9, and 2 for USAGE=A2, for a total of 14.

Example: Accessing a Group Field With Multiple Formats

When you use a group field with multiple formats in a query, you must account for each position in the group, including trailing blanks or leading zeros. The following example illustrates how to access a group field with multiple formats in a query:

```
GROUP = GRPB, ALIAS = KEY, USAGE = A8, ACTUAL = A8 ,$
 FIELDNAME = FIELD1, ALIAS = F1, USAGE = A2, ACTUAL = A2 ,$
 FIELDNAME = FIELD2, ALIAS = F2, USAGE = I8, ACTUAL = I4 ,$
 FIELDNAME = FIELD3, ALIAS = F3, USAGE = A2, ACTUAL = A2 ,$
```

The values in fields F1 and F3 may include some trailing blanks, and the values in field F2 may include some leading zeros. When using the group in a query, you must account for each position. Because FIELD2 is a numeric field, you cannot specify the IF criteria as follows:

```
IF GRPB EQ 'A 0334BB'
```

You can eliminate this error by using a slash (/) to separate the components of the group key:

```
IF GRPB EQ 'A/334/BB'
```

Note: Blanks and leading zeros are assumed where needed to fill out the key.

Describe these multiply occurring fields by placing them in a separate segment. Fields A and B are placed in the root segment. Fields C1 and C2, which occur multiply in relation to A and B, are placed in a descendant segment. You use an additional segment attribute, the OCCURS attribute, to specify that these segments represent multiply occurring fields. In certain cases, you may also need a second attribute, called the POSITION attribute.

Using the OCCURS Attribute

The OCCURS attribute is an optional segment attribute used to describe records containing repeating fields or groups of fields. Define such records by describing the singly occurring fields in one segment, and the multiply occurring fields in a descendant segment. The OCCURS attribute appears in the declaration for the descendant segment.

You can have several sets of repeating fields in your data structure. Describe each of these sets of fields as a separate segment in your data source description. Sets of repeating fields can be divided into two basic types: parallel and nested.

Syntax: How to Specify a Repeating Field

```
OCCURS = occurstype
```

Possible values for *occurstype* are:

n

Is an integer value showing the number of occurrences (from 1 to 4095).

fieldname

Names a field in the parent segment whose integer value contains the number of occurrences of the descendant segment.

VARIABLE

Indicates that the number of occurrences varies from record to record. The number of occurrences is computed from the record length (for example, if the field lengths for the segment add up to 40, and 120 characters are read in, it means there are three occurrences).

Place the OCCURS attribute in your segment declaration after the PARENT attribute.

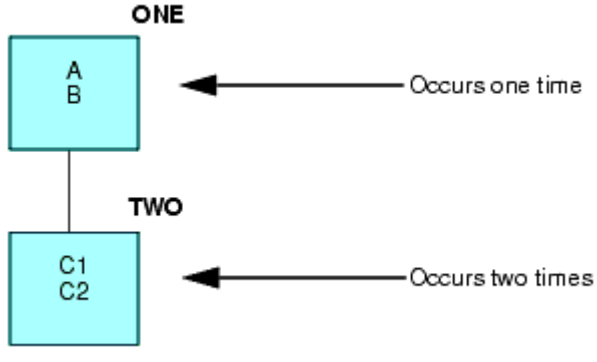
When different types of records are combined in one data source, each record type can contain only one segment defined as OCCURS=VARIABLE. It may have OCCURS descendants (if it contains a nested group), but it may not be followed by any other segment with the same parent, that is, there can be no other segments to its right in the hierarchical data structure. This restriction is necessary to ensure that data in the record is interpreted unambiguously.

Example: **Using the OCCURS Attribute**

Consider the following simple data structure:

A	B	C1	C2	C1	C2

You have two occurrences of fields C1 and C2 for every one occurrence of fields A and B. Thus, to describe this data source, you place fields A and B in the root segment, and fields C1 and C2 in a descendant segment, as shown here:



Describe this data source as follows:

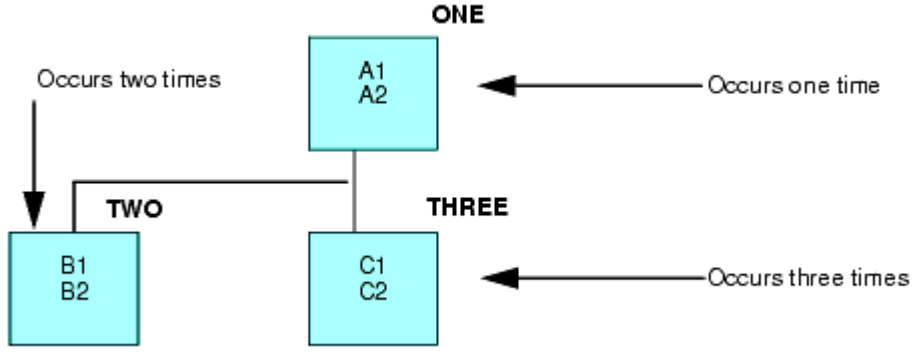
```
FILENAME = EXAMPLE1, SUFFIX = FIX, $
 SEGNAME = ONE, SEGTYPE=S0, $
  FIELDNAME = A,  ALIAS=, USAGE = A2, ACTUAL = A2, $
  FIELDNAME = B,  ALIAS=, USAGE = A1, ACTUAL = A1, $
 SEGNAME = TWO, PARENT = ONE, OCCURS = 2, SEGTYPE=S0, $
  FIELDNAME = C1, ALIAS=, USAGE = I4, ACTUAL = I2, $
  FIELDNAME = C2, ALIAS=, USAGE = I4, ACTUAL = I2, $
```

Describing a Parallel Set of Repeating Fields

Parallel sets of repeating fields are those that have nothing to do with one another (that is, they have no parent-child or logical relationship). Consider the following data structure:

A1	A2	B1	B2	B1	B2	C1	C2	C1	C2	C1	C2

In this example, fields B1 and B2 and fields C1 and C2 repeat within the record. The number of times that fields B1 and B2 occur has nothing to do with the number of times fields C1 and C2 occur. Fields B1 and B2 and fields C1 and C2 are parallel sets of repeating fields. They should be described in the data source description as children of the same parent, the segment that contains fields A1 and A2. The following data structure reflects their relationship.

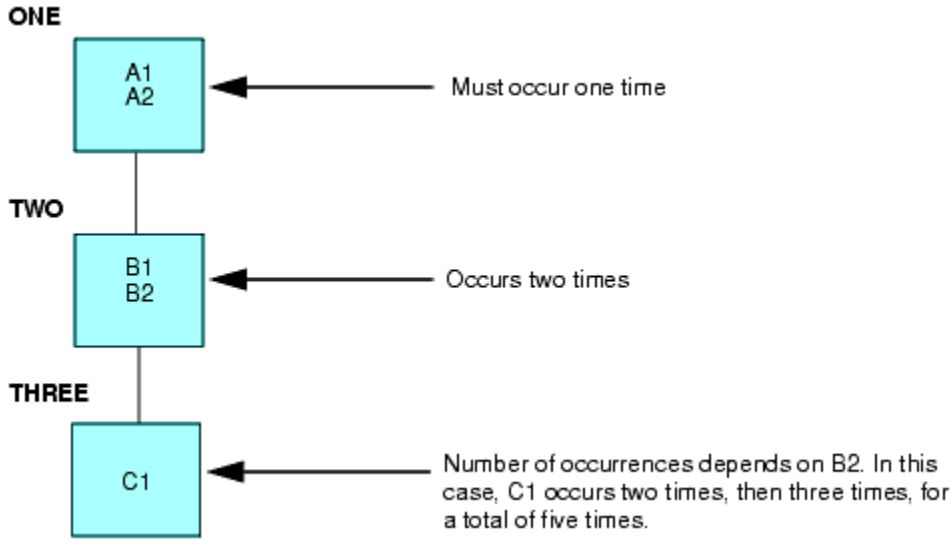


Describing a Nested Set of Repeating Fields

Nested sets of repeating fields are those whose occurrence depends on one another in some way. Consider the following data structure:

A1	A2	B1	B2	C1	C1	B1	B2	C1	C1	C1

In this example, field C1 only occurs after fields B1 and B2 occur once. It occurs varying numbers of times, recorded by a counter field, B2. There is not a set of occurrences of C1 which is not preceded by an occurrence of fields B1 and B2. Fields B1, B2, and C1 are a nested set of repeating fields. They can be represented by the following data structure:



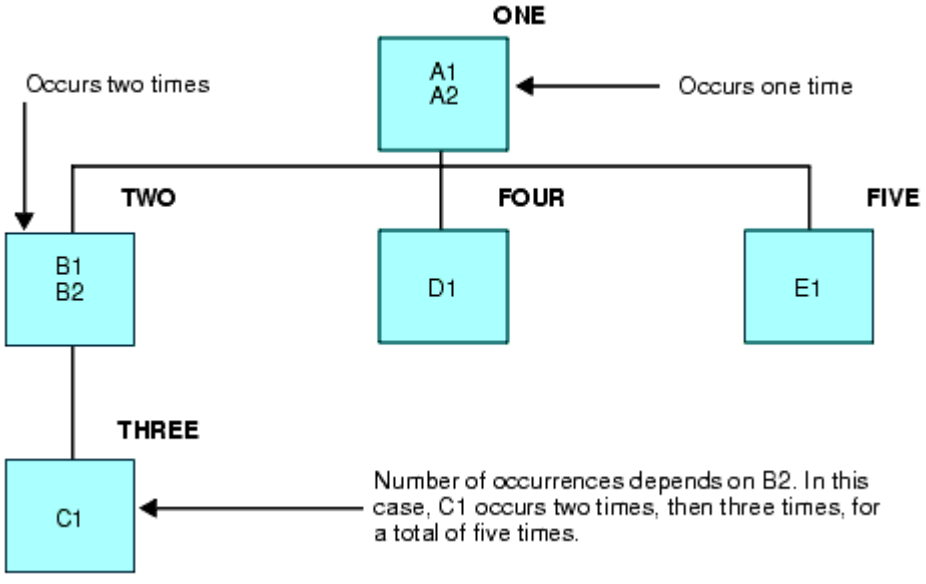
Since field C1 repeats with relation to fields B1 and B2, which repeat in relation to fields A1 and A2, field C1 is described as a separate, descendant segment of Segment TWO, which is in turn a descendant of Segment ONE.

Example: Describing Parallel and Nested Repeating Fields

The following data structure contains both nested and parallel sets of repeating fields.

A 1	A 2	B 1	B 2	C 1	C 1	C 1	B 1	B 2	C 1	C 1	C 1	C 1	D 1	D 1	E 1	E 1	E 1	E 1

It produces the following data structure:



Describe this data source as follows. Notice that the assignment of the PARENT attributes shows you how the occurrences are nested.

```
FILENAME = EXAMPLE3, SUFFIX = FIX,$
 SEGNAME = ONE,    SEGTYPE=S0,$
  FIELDNAME = A1 ,ALIAS= ,ACTUAL = A1  ,USAGE = A1  ,$
  FIELDNAME = A2 ,ALIAS= ,ACTUAL = I1  ,USAGE = I1  ,$
 SEGNAME = TWO,    SEGTYPE=S0, PARENT = ONE, OCCURS = 2  ,$
  FIELDNAME = B1 ,ALIAS= ,ACTUAL = A15 ,USAGE = A15 ,$
  FIELDNAME = B2 ,ALIAS= ,ACTUAL = I1  ,USAGE = I1  ,$
 SEGNAME = THREE, SEGTYPE=S0, PARENT = TWO, OCCURS = B2 ,$
  FIELDNAME = C1 ,ALIAS= ,ACTUAL = A25 ,USAGE = A25 ,$
 SEGNAME = FOUR,  SEGTYPE=S0, PARENT = ONE, OCCURS = A2 ,$
  FIELDNAME = D1 ,ALIAS= ,ACTUAL = A15 ,USAGE = A15 ,$
 SEGNAME = FIVE,  SEGTYPE=S0, PARENT = ONE, OCCURS = VARIABLE,$
  FIELDNAME = E1 ,ALIAS= ,ACTUAL = A5  ,USAGE = A5  ,$
```

Note:

- Segments ONE, TWO, and THREE represent a nested group of repeating segments. Fields B1 and B2 occur a fixed number of times, so OCCURS equals 2. Field C1 occurs a certain number of times within each occurrence of fields B1 and B2. The number of times C1 occurs is determined by the value of field B2, which is a counter. In this case, its value is 3 for the first occurrence of Segment TWO and 4 for the second occurrence.

- Segments FOUR and FIVE consist of fields that repeat independently within the parent segment. They have no relationship to each other or to Segment TWO except their common parent, so they represent a parallel group of repeating segments.
- As in the case of Segment THREE, the number of times Segment FOUR occurs is determined by a counter in its parent, A2. In this case, the value of A2 is two.
- The number of times Segment FIVE occurs is variable. This means that all the rest of the fields in the record (all those to the right of the first occurrence of E1) are read as recurrences of field E1. To ensure that data in the record is interpreted unambiguously, a segment defined as OCCURS=VARIABLE must be at the end of the record. In a data structure diagram, it is the rightmost segment. Note that there can be only one segment defined as OCCURS=VARIABLE for each record type.

Using the POSITION Attribute

The POSITION attribute is an optional attribute used to describe a structure in which multiply occurring fields with an established number of occurrences are located in the middle of the record. You describe the data source as a hierarchical structure, made up of a parent segment and at least one child segment that contains the multiply occurring fields. The parent segment is made up of whatever singly occurring fields are in the record, as well as one or more alphanumeric fields that appear where the multiply occurring fields appear in the record. The alphanumeric field may be a dummy field that is the exact length of the combined multiply occurring fields. For example, if you have four occurrences of an 8-character field, the length of the field in the parent segment is 32 characters.

You can also use the POSITION attribute to re-describe fields with SEGTYPE=U. See *Redefining a Field in a VSAM Data Source* on page 2541.

Syntax: How to Specify the Position of a Repeating Field

The POSITION attribute is described in the child segment. It gives the name of the field in the parent segment that specifies the starting position and overall length of the multiply occurring fields. The syntax of the POSITION attribute is

```
POSITION = fieldname
```

where:

fieldname

Is the name of the field in the parent segment that defines the starting position of the multiple field occurrences.

Example: **Specifying the Position of a Repeating Field**

Consider the following data structure:

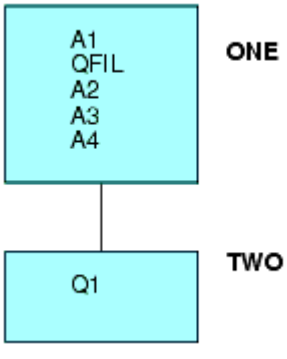
A1	Q1	Q1	Q1	Q1	A2	A3	A4

In this example, field Q1 repeats four times in the middle of the record. When you describe this structure, you specify a field or fields that occupy the position of the four Q1 fields in the record. You then assign the actual Q1 fields to a multiply occurring descendant segment. The POSITION attribute, specified in the descendant segment, gives the name of the field in the parent segment that identifies the starting position and overall length of the Q fields.

Use the following Master File to describe this structure:

```
FILENAME = EXAMPLE3, SUFFIX = FIX,$
 SEGNAME = ONE, SEGTYPE=S0,$
  FIELDNAME = A1    ,ALIAS= ,USAGE = A14 ,ACTUAL = A14 ,$
  FIELDNAME = QFIL  ,ALIAS= ,USAGE = A32 ,ACTUAL = A32 ,$
  FIELDNAME = A2    ,ALIAS= ,USAGE = I2  ,ACTUAL = I2  ,$
  FIELDNAME = A3    ,ALIAS= ,USAGE = A10 ,ACTUAL = A10 ,$
  FIELDNAME = A4    ,ALIAS= ,USAGE = A15 ,ACTUAL = A15 ,$
 SEGNAME = TWO, SEGTYPE=S0, PARENT = ONE, POSITION = QFIL, OCCURS = 4 ,$
  FIELDNAME = Q1    ,ALIAS= ,USAGE = D8  ,ACTUAL = D8  ,$
```

This produces the following structure:



If the total length of the multiply occurring fields is longer than 4095, you can use a filler field after the dummy field to make up the remaining length. This is required, because the format of an alphanumeric field cannot exceed 4095 bytes.

Notice that this structure works only if you have a fixed number of occurrences of the repeating field. This means the OCCURS attribute of the descendant segment must be of the type OCCURS=*n*. OCCURS=*fieldname* or OCCURS=VARIABLE does not work.

Specifying the ORDER Field

In an OCCURS segment, the order of the data may be significant. For example, the values may represent monthly or quarterly data, but the record itself may not explicitly specify the month or quarter to which the data applies.

To associate a sequence number with each occurrence of the field, you may define an internal counter field in any OCCURS segment. A value is automatically supplied that defines the sequence number of each repeating group.

Syntax: How to Specify the Sequence of a Repeating Field

The syntax rules for an ORDER field are:

- It must be the last field described in an OCCURS segment.
- The field name is arbitrary.
- The ALIAS is ORDER.
- The USAGE is I*n*, with any appropriate edit options.
- The ACTUAL is I4.

For example:

```
FIELD = ACT_MONTH, ALIAS = ORDER, USAGE = I2MT, ACTUAL = I4, $
```

Order values are 1, 2, 3, and so on, within each occurrence of the segment. The value is assigned prior to any selection tests that might accept or reject the record, and so it can be used in a selection test.

For example, to obtain data for only the month of June, type:

```
SUM AMOUNT...
WHERE ACT_MONTH IS 6
```

The ORDER field is a virtual field used internally. It does not alter the logical record length (LRECL) of the data source being accessed.

Redefining a Field in a VSAM Data Source

Redefining record fields in non-FOCUS data sources is supported. This enables you to describe a field with an alternate layout.

Within the Master File, the redefined fields must be described in a separate unique segment (SEGTYPE=U) using the POSITION=*fieldname* and OCCURS=1 attributes.

The redefined fields can have any user-defined name.

Syntax: How to Redefine a Field

```
SEGNAME = segname, SEGTYPE = U, PARENT = parentseg,
OCCURS = 1, POSITION = fieldname,$
```

where:

segname

Is the name of the segment.

parentseg

Is the name of the parent segment.

fieldname

Is the name of the first field being redefined. Using the unique segment with redefined fields helps avoid problems with multipath reporting.

A one-to-one relationship forms between the parent record and the redefined segment.

Example: Redefining a VSAM Structure

The following example illustrates redefinition of the VSAM structure described in the COBOL file description, where the COBOL FD is:

```
01 ALLFIELDS
  02 FLD1   PIC X(4)         -  this describes alpha/numeric data
  02 FLD2   PIC X(4)         -  this describes numeric data
  02 RFLD1  PIC 9(5)V99 COMP-3 REDEFINES FLD2
  02 FLD3   PIC X(8)         -  this describes alpha/numeric data

FILE = REDEF, SUFFIX = VSAM,$
 SEGNAME = ONE, SEGTYPE = S0,$
  GROUP = RKEY, ALIAS = KEY, USAGE = A4 ,ACTUAL = A4 ,$
   FIELDNAME = FLD1,,  USAGE = A4   ,ACTUAL = A4 ,$
   FIELDNAME = FLD2,,  USAGE = A4   ,ACTUAL = A4 ,$
   FIELDNAME = FLD3,,  USAGE = A8   ,ACTUAL = A8 ,$
 SEGNAME = TWO, SEGTYPE = U, POSITION = FLD2, OCCURS = 1, PARENT = ONE ,$
   FIELDNAME = RFLD1,, USAGE = P8.2 ,ACTUAL = Z4 ,$
```

Reference: Special Considerations for Redefining a Field

- Redefinition is a read-only feature and is used only for presenting an alternate view of the data. It is not used for changing the format of the data.

- For non-alphanumeric fields, you must know your data. Attempts to print numeric fields that contain alphanumeric data produce data exceptions or errors converting values. It is recommended that the first definition always be alphanumeric to avoid conversion errors.
- More than one field can be redefined in a segment.
- Redefines are supported only for IDMS, IMS, VSAM, Db2, and FIX data sources.

Extra-Large Record Length Support With VSAM

If the Master File describes a data source with OCCURS segments, and if the longest single record in the data source is larger than 16K bytes, it is necessary to specify a larger record size in advance.

Syntax: How to Define the Maximum Record Length

```
SET MAXLRECL = nnnnn
```

where:

nnnnn

Is up to 32768.

For example, SET MAXLRECL=12000 allows handling of records that are 12000 bytes long. Once you have entered the SET MAXLRECL command, you can obtain the current value of the MAXLRECL parameter by using the ? SET MAXLRECL command.

If the actual record length is longer than specified, retrieval halts and the actual record length appears in hexadecimal notation.

Describing Multiple Record Types in VSAM Data Sources

VSAM data sources can contain more than one type of record. When they do, they can be structured in one of two ways:

- A positional relationship may exist between the various record types, with a record of one type being followed by one or more records containing detailed information about the first record.

  If a positional relationship exists between the various record types, with a parent record of one type followed by one or more child records containing detail information about the parent, you describe the structure by defining the parent as the root, and the detail segments as descendants.

Some VSAM data sources are structured so that descendant records relate to each other through concatenating key fields. That is, the key fields of a parent record serve as the first part of the key of a child record. In such cases, the segment key fields must be described using a GROUP declaration. The GROUP key fields of each segment consist of the renamed key fields from the parent segment plus the unique key field from the child record.

- The records have no meaningful positional relationship, and records of varying types exist independently of each other in the data source.

  If the records have no meaningful positional relationship, you have to provide some means for interpreting the type of record that has been read. Do this by creating a dummy root segment for the records.

Key-sequenced VSAM data sources also use the RECTYPE attribute to distinguish various record types within the data source.

A parent does not always share its RECTYPE with its descendants. It may share some other identifying piece of information, such as the PUBNO in the example. This is the field that should be included in the parent key, as well as all of its descendants keys, to relate them.

When using the RECTYPE attribute in VSAM data sources with group keys, the RECTYPE field can be part of the segment's group key only when it belongs to a segment that has no descendants, or to a segment whose descendants are described with an OCCURS attribute. In *Describing VSAM Positionally Related Records* on page 2549, the RECTYPE field is added to the group key in the SERIANO segment, the lowest descendant segment in the chain.

Describing a RECTYPE Field

When a data source contains multiple record types, there must be a field in the records themselves that can be used to differentiate between the record types. You can find information on this field in your existing description of the data source (for example, a COBOL FD statement). This field must appear in the same physical location of each record. For example, columns 79 and 80 can contain a different 2-digit code for each unique record type. Describe this identifying field with the field name RECTYPE.

Another technique for redefining the parts of records is to use the MAPFIELD and MAPVALUE attributes described in *Describing a Repeating Group Using MAPFIELD* on page 2561.

Syntax: How to Specify a Record Type Field

The RECTYPE field must fall in the same physical location of each record in the data source, or the record is ignored. The syntax to describe the RECTYPE field is

```
FIELDNAME = RECTYPE, ALIAS = value, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

value

Is the record type in alphanumeric format, if an ACCEPT list is not specified. If there is an ACCEPT list, this can be any value.

format

Is the data type of the field. In addition to RECTYPE fields in alphanumeric format, RECTYPE fields in packed and integer formats (formats P and I) are supported. Possible values are:

A*n*

Where *n* is 1-4095. Indicates character data, including letters, digits, and other characters.

I*n*

Indicates ACTUAL (internal) format binary integers:

- I1 = single-byte binary integer.
- I2 = half-word binary integer (2 bytes).
- I4 = full-word binary integer (4 bytes).

The USAGE format can be I1 through I9, depending on the magnitude of the ACTUAL format.

P*n*

Where *n* is 1-16. Indicates packed decimal ACTUAL (internal) format. n is the number of bytes, each of which contains two digits, except for the last byte which contains a digit and the sign. For example, P6 means 11 digits plus a sign.

If the field contains an assumed decimal point, represent the field with a USAGE format of P*m.n*, where *m* is the total number of digits, and *n* is the number of decimal places. Thus, P11.1 means an 11-digit number with one decimal place.

list

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Separate each item in the list with either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks. The list may contain a single RECTYPE value. For example:

```
FIELDNAME = RECTYPE, ALIAS = TYPEABC, USAGE = A1,
ACTUAL = A1, ACCEPT = A OR B OR C, $
```

range

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed within single quotation marks.

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order. For example:

```
FIELDNAME = RECTYPE, ALIAS = ACCTREC, USAGE = P3,
ACTUAL = P2, ACCEPT = 100 TO 200, $
```

Example: Specifying the RECTYPE Field

The following field description is of a 1 byte packed RECTYPE field containing the value 1:

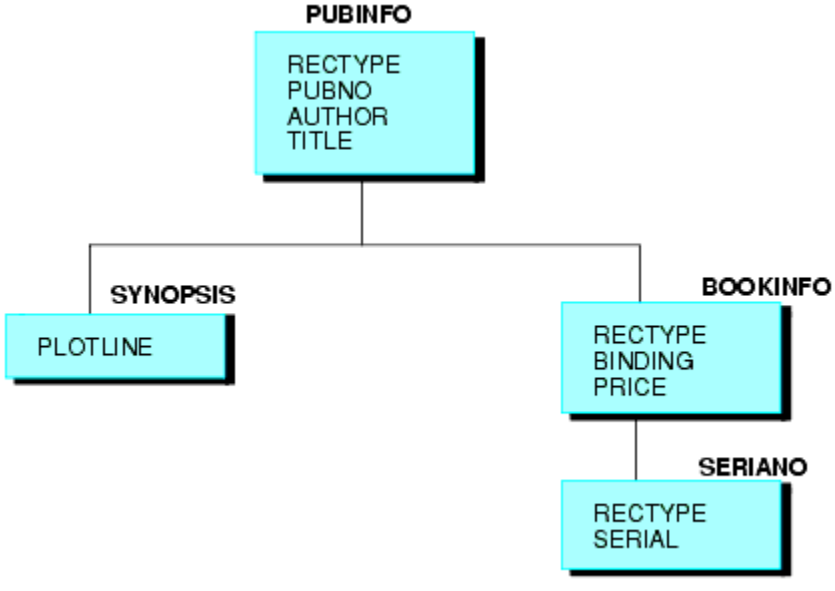
```
FIELD = RECTYPE, ALIAS = 1, USAGE = P1, ACTUAL = P1, $
```

The following field description is of a 3 byte alphanumeric RECTYPE field containing the value A34:

```
FIELD = RECTYPE, ALIAS = A34, USAGE = A3, ACTUAL = A3,$
```

Describing Positionally Related Records

The following diagram shows a more complex version of the library data source:



Information that is common to all copies of a given book (the identifying number, the author's name, and its title) has the same record type. All of this information is assigned to the root segment in the Master File. The synopsis is common to all copies of a given book, but in this data source it is described as a series of repeating fields of ten characters each, in order to save space.

The synopsis is assigned to its own subordinate segment with an attribute of OCCURS=VARIABLE in the Master File. Although there are segments in the diagram to the right of the OCCURS=VARIABLE segment, OCCURS=VARIABLE is the rightmost segment within its own record type. Only segments with a RECTYPE that is different from the OCCURS=VARIABLE segment can appear to its right in the structure. Note also that the OCCURS=VARIABLE segment does not have a RECTYPE. This is because it is part of the same record as its parent segment.

Binding and price can vary among copies of a given title. For instance, the library may have two different versions of *Pamela*, one a paperback costing $7.95, the other a hardcover costing $15.50. These two fields are of a second record type, and are assigned to a descendant segment in the Master File.

Finally, every copy of the book in the library has its own identifying serial number, which is described in a field of record type S. In the Master File, this information is assigned to a segment that is a child of the segment containing the binding and price information.

Use the following Master File to describe this data source:

```
FILENAME = LIBRARY2, SUFFIX = FIX,$
 SEGNAME = PUBINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = P     ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = PUBNO    ,ALIAS = PN    ,USAGE = A10   ,ACTUAL = A10 ,$
  FIELDNAME = AUTHOR   ,ALIAS = AT    ,USAGE = A25   ,ACTUAL = A25 ,$
  FIELDNAME = TITLE    ,ALIAS = TL    ,USAGE = A50   ,ACTUAL = A50 ,$
 SEGNAME = SYNOPSIS, PARENT = PUBINFO,  OCCURS = VARIABLE, SEGTYPE = S0,$
  FIELDNAME = PLOTLINE ,ALIAS = PLOTL ,USAGE = A10   ,ACTUAL = A10 ,$
 SEGNAME = BOOKINFO, PARENT = PUBINFO,  SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = B     ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = BINDING  ,ALIAS = BI    ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = PRICE    ,ALIAS = PR    ,USAGE = D8.2N ,ACTUAL = D8  ,$
 SEGNAME = SERIANO,  PARENT = BOOKINFO, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = S     ,USAGE = A1    ,ACTUAL = A1  ,$
  FIELDNAME = SERIAL   ,ALIAS = SN    ,USAGE = A15   ,ACTUAL = A15 ,$
```

Note that each segment, except OCCURS, contains a field named RECTYPE and that the ALIAS for the field contains a unique value for each segment (P, B, and S). If there is a record in this data source with a RECTYPE other than P, B, or S, the record is ignored. The RECTYPE field must fall in the same physical location in each record.

Ordering of Records in the Data Source

Physical order determines parent/child relationships in sequential records. Every parent record does not need descendants. Specify how you want data in missing segment instances handled in your reports by using the SET command to change the ALL parameter.

In the example in *Describing Positionally Related Records* on page 2547, if the first record in the data source is not a PUBINFO record, the record is considered to be a child without a parent. Any information allotted to the SYNOPSIS segment appears in the PUBINFO record. The next record may be a BOOKINFO or even another PUBINFO (in which case the first PUBINFO is assumed to have no descendants). Any SERIANO records are assumed to be descendants of the previous BOOKINFO record. If a SERIANO record follows a PUBINFO record with no intervening BOOKINFO, it is treated as if it has no parent.

Example: Describing VSAM Positionally Related Records

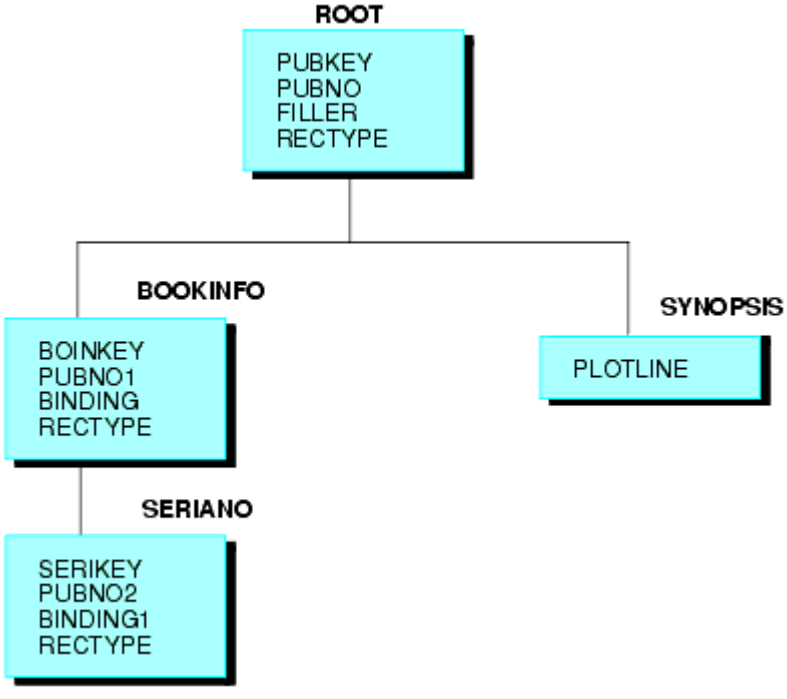
Consider the following VSAM data source that contains three types of records. The ROOT records have a key that consists of the publisher's number, PUBNO. The BOOKINFO segment has a key that consists of that same publisher number, plus a hard-cover or soft-cover indicator, BINDING. The SERIANO segment key consists of the first two elements, plus a record type field, RECTYPE.

```
FILENAME = LIBRARY6, SUFFIX = VSAM,$
 SEGNAME = ROOT, SEGTYPE = S0,$
  GROUP=PUBKEY        ,ALIAS=KEY   ,USAGE=A10   ,ACTUAL=A10  ,$
   FIELDNAME=PUBNO    ,ALIAS=PN    ,USAGE=A10   ,ACTUAL=A10  ,$
  FIELDNAME=FILLER    ,ALIAS=      ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE   ,ALIAS=1     ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=AUTHOR    ,ALIAS=AT    ,USAGE=A25   ,ACTUAL=A25  ,$
  FIELDNAME=TITLE     ,ALIAS=TL    ,USAGE=A50   ,ACTUAL=A50  ,$
 SEGNAME=BOOKINFO,PARENT=ROOT,   SEGTYPE=S0,$
  GROUP=BOINKEY       ,ALIAS=KEY   ,USAGE=A11   ,ACTUAL=A11  ,$
   FIELDNAME=PUBNO1   ,ALIAS=P1    ,USAGE=A10   ,ACTUAL=A10  ,$
   FIELDNAME=BINDING  ,ALIAS=BI    ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=RECTYPE   ,ALIAS=2     ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=PRICE     ,ALIAS=PR    ,USAGE=D8.2N ,ACTUAL=D8   ,$
 SEGNAME=SERIANO, PARENT=BOOKINFO,SEGTYPE=S0,$
  GROUP=SERIKEY       ,ALIAS=KEY   ,USAGE=A12   ,ACTUAL=A12  ,$
   FIELDNAME=PUBNO2   ,ALIAS=P2    ,USAGE=A10   ,ACTUAL=A10  ,$
   FIELDNAME=BINDING1 ,ALIAS=B1    ,USAGE=A1    ,ACTUAL=A1   ,$
   FIELDNAME=RECTYPE  ,ALIAS=3     ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=SERIAL    ,ALIAS=SN    ,USAGE=A15   ,ACTUAL=A15  ,$
 SEGNAME=SYNOPSIS,PARENT=ROOT,    SEGTYPE=S0, OCCURS=VARIABLE,$
  FIELDNAME=PLOTLINE  ,ALIAS=PLOTL ,USAGE=A10   ,ACTUAL=A10  ,$
```

Notice that the length of the key fields specified in the USAGE and ACTUAL attributes of a GROUP declaration is the length of the key fields from the parent segments, plus the length of the added field of the child segment (RECTYPE field). In the example above, the length of the GROUP key SERIKEY equals the length of PUBNO2 and BINDING1, the group key from the parent segment, plus the length of RECTYPE, the field added to the group key in the child segment. The length of the key increases as you traverse the structure.

Note: Each segment key describes as much of the true key as needed to find the next instance of that segment.

In the sample data source, the repetition of the publisher number as PUBNO1 and PUBNO2 in the descendant segments interrelates the three types of records. The data source can be diagrammed as the following structure:



A typical query may request information on price and call numbers for a specific publisher's number:

```
PRINT PRICE AND SERIAL BY PUBNO
IF PUBNO EQ 1234567890 OR 9876054321
```

Since PUBNO is part of the key, retrieval can occur quickly, and the processing continues. To further speed retrieval, add search criteria based on the BINDING field, which is also part of the key.

Describing Unrelated Records

Some VSAM data sources do not have records that are related to one another. That is, the VSAM key of one record type is independent of the keys of other record types. To describe data sources with unrelated records, define a dummy root segment for the record types. The following rules apply to the dummy root segment:

- The name of the root segment must be DUMMY.
- It must have only one field with a blank name and alias.
- The USAGE and ACTUAL attributes must both be A1.

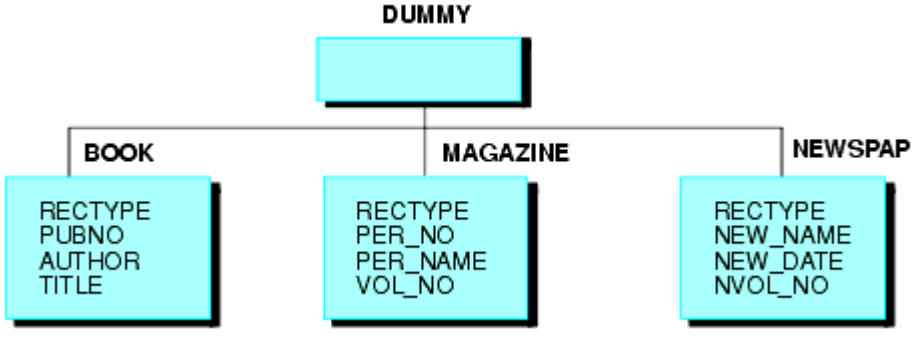
All other non-repeating segments must point to the dummy root as their parent. Except for the root, all non-repeating segments must have a RECTYPE and a PARENT attribute and describe the full VSAM key. If the data source does not have a key, the group should not be described. RECTYPEs may be anywhere in the record.

Example: Describing Unrelated Records Using a Dummy Root Segment

The library data source has three types of records: book information, magazine information, and newspaper information. Since these three record types have nothing in common, they cannot be described as parent records followed by detail records.

The data source can look like this:

A structure such as the following can also describe this data source:



The Master File for the structure in this example is:

```
FILENAME = LIBRARY3, SUFFIX = FIX,$
 SEGMENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME=           ,ALIAS=     ,USAGE = A1    ,ACTUAL = A1   ,$
 SEGMENT = BOOK, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = B  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PUBNO    ,ALIAS = PN ,USAGE = A10   ,ACTUAL = A10  ,$
  FIELDNAME = AUTHOR   ,ALIAS = AT ,USAGE = A25   ,ACTUAL = A25  ,$
  FIELDNAME = TITLE    ,ALIAS = TL ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = BINDING  ,ALIAS = BI ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PRICE    ,ALIAS = PR ,USAGE = D8.2N ,ACTUAL = D8   ,$
  FIELDNAME = SERIAL   ,ALIAS = SN ,USAGE = A15   ,ACTUAL = A15  ,$
  FIELDNAME = SYNOPSIS ,ALIAS = SY ,USAGE = A150  ,ACTUAL = A150 ,$
 SEGMENT = MAGAZINE, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = M  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = PER_NO   ,ALIAS = PN ,USAGE = A10   ,ACTUAL = A10  ,$
  FIELDNAME = PER_NAME ,ALIAS = NA ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = VOL_NO   ,ALIAS = VN ,USAGE = I2    ,ACTUAL = I2   ,$
  FIELDNAME = ISSUE_NO ,ALIAS = IN ,USAGE = I2    ,ACTUAL = I2   ,$
  FIELDNAME = PER_DATE ,ALIAS = DT ,USAGE = I6MDY ,ACTUAL = I6   ,$
 SEGMENT = NEWSPAP, PARENT = DUMMY, SEGTYPE = S0,$
  FIELDNAME = RECTYPE  ,ALIAS = N  ,USAGE = A1    ,ACTUAL = A1   ,$
  FIELDNAME = NEW_NAME ,ALIAS = NN ,USAGE = A50   ,ACTUAL = A50  ,$
  FIELDNAME = NEW_DATE ,ALIAS = ND ,USAGE = I6MDY ,ACTUAL = I6   ,$
  FIELDNAME = NVOL_NO  ,ALIAS = NV ,USAGE = I2    ,ACTUAL = I2   ,$
  FIELDNAME = ISSUE    ,ALIAS = NI ,USAGE = I2    ,ACTUAL = I2   ,$
```

Example: Describing a VSAM Data Source With Unrelated Records

Consider another VSAM data source containing information on the library. This data source has three types of records: book information, magazine information, and newspaper information.

There are two possible structures:

- The RECTYPE is the beginning of the key. The key structure is:

RECTYPE B	Book Code
RECTYPE M	Magazine Code
RECTYPE N	Newspaper Code

The sequence of records is:

Book

Book

Magazine

Magazine

Newspaper

Newspaper

Note the difference between the use of the RECTYPE here and its use when the records are positionally related. In this case, the codes are unrelated and the database designer has chosen to accumulate the records by type first (all the book information together, all the magazine information together, and all the newspaper information together), so the RECTYPE may be the initial part of the key.

- The RECTYPE is not in the beginning of the key or is outside of the key. The key structure is:

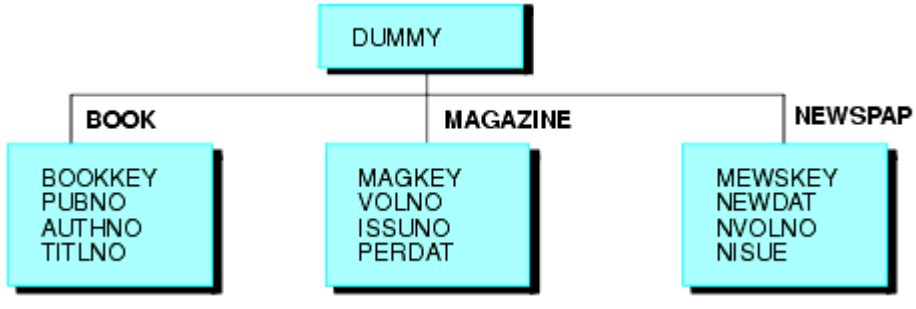
Book Code

Magazine Code

Newspaper Code

The sequence of record types in the data source can be arbitrary.

Both types of file structure can be represented by the following:



Example: Describing a Key and a Record Type for a VSAM Data Source With Unrelated Records

```
FILE=LIBRARY7, SUFFIX=VSAM,$
 SEGMENT=DUMMY,$
  FIELDNAME=          ,ALIAS=    ,USAGE=A1    ,ACTUAL=A1   ,$
 SEGMENT=BOOK,      PARENT=DUMMY,SEGTYPE=S0,$
  GROUP=BOOKKEY       ,ALIAS=KEY ,USAGE=A11   ,ACTUAL=A11  ,$
   FIELDNAME=PUBNO    ,ALIAS=PN  ,USAGE=A3    ,ACTUAL=A3   ,$
   FIELDNAME=AUTHNO   ,ALIAS=AN  ,USAGE=A3    ,ACTUAL=A3   ,$
   FIELDNAME=TITLNO   ,ALIAS=TN  ,USAGE=A4    ,ACTUAL=A4   ,$
   FIELDNAME=RECTYPE  ,ALIAS=B   ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=AUTHOR    ,ALIAS=AT  ,USAGE=A25   ,ACTUAL=A25  ,$
  FIELDNAME=TITLE     ,ALIAS=TL  ,USAGE=A50   ,ACTUAL=A50  ,$
  FIELDNAME=BINDING   ,ALIAS=BI  ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=PRICE     ,ALIAS=PR  ,USAGE=D8.2N ,ACTUAL=D8   ,$
  FIELDNAME=SERIAL    ,ALIAS=SN  ,USAGE=A15   ,ACTUAL=A15  ,$
  FIELDNAME=SYNOPSIS ,ALIAS=SY  ,USAGE=A150  ,ACTUAL=A150 ,$
 SEGMENT=MAGAZINE, PARENT=DUMMY, SEGTYPE=S0,$
  GROUP=MAGKEY        ,ALIAS=KEY ,USAGE=A11   ,ACTUAL=A11  ,$
   FIELDNAME=VOLNO    ,ALIAS=VN  ,USAGE=A2    ,ACTUAL=A2   ,$
   FIELDNAME=ISSUNO   ,ALIAS=IN  ,USAGE=A2    ,ACTUAL=A2   ,$
   FIELDNAME=PERDAT   ,ALIAS=DT  ,USAGE=A6    ,ACTUAL=A6   ,$
   FIELDNAME=RECTYPE  ,ALIAS=M   ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=PER_NAME ,ALIAS=PRN ,USAGE=A50   ,ACTUAL=A50  ,$
 SEGMENT=NEWSPAP,   PARENT=DUMMY, SEGTYPE=S0,$
  GROUP=NEWSKEY       ,ALIAS=KEY ,USAGE=A11   ,ACTUAL=A11  ,$
   FIELDNAME=NEWDAT   ,ALIAS=ND  ,USAGE=A6    ,ACTUAL=A6   ,$
   FIELDNAME=NVOLNO   ,ALIAS=NV  ,USAGE=A2    ,ACTUAL=A2   ,$
   FIELDNAME=NISSUE   ,ALIAS=NI  ,USAGE=A2    ,ACTUAL=A2   ,$
   FIELDNAME=RECTYPE  ,ALIAS=N   ,USAGE=A1    ,ACTUAL=A1   ,$
  FIELDNAME=NEWNAME   ,ALIAS=NN  ,USAGE=A50   ,ACTUAL=A50  ,$
```

Using a Generalized Record Type

If your VSAM data source has multiple record types that share the same layout, you can specify a single generalized segment that describes all record types that have the common layout. By using a generalized segment, also known as a generalized RECTYPE, instead of one segment per record type, you reduce the number of segments you need to describe in the Master File.

When using a generalized segment, you identify RECTYPE values using the ACCEPT attribute. You can assign any value to the ALIAS attribute.

Syntax: How to Specify a Generalized Record Type

```
FIELDNAME = RECTYPE, ALIAS = alias, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

`RECTYPE`

Is the required field name.

Note: Since the field name, RECTYPE, may not be unique across segments, you should not use it in this way unless you qualify it. An alias is not required; you may leave it blank.

`alias`

Is any valid alias specification. You can specify a unique name as the alias value for the RECTYPE field only if you use the ACCEPT attribute. The alias can then be used in a TABLE request as a display field, a sort field, or in selection tests using either WHERE or IF.

`list`

Is a list of one or more lines of specific RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks. The list may contain a single RECTYPE value. For example:

```
FIELDNAME = RECTYPE, ALIAS = TYPEABC, USAGE = A1,
ACTUAL = A1, ACCEPT = A OR B OR C, $
```

`range`

Is a range of one or more lines of RECTYPE values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed within single quotation marks.

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order. For example:

```
FIELDNAME = RECTYPE, ALIAS = ACCTREC, USAGE = P3,
ACTUAL = P2, ACCEPT = 100 TO 200, $
```

Example: Using a Generalized Record Type

To illustrate the use of the generalized record type in a VSAM Master File, consider the following record layouts in the DOC data source. Record type DN is the root segment and contains the document number and title. Record types M, I, and C contain information about manuals, installation guides, and course guides, respectively. Notice that record types M and I have the same layout.

Record type DN:

```
---KEY---
+-------------------------------------------------------------------------+
DOCID   FILLER          RECTYPE     TITLE
+-------------------------------------------------------------------------+
```

Record type M:

```
--------KEY--------
+-------------------------------------------------------------------------+
MDOCID   MDATE          RECTYPE     MRELEASE         MPAGES     FILLER
+-------------------------------------------------------------------------+
```

Record type I:

```
--------KEY--------
+-------------------------------------------------------------------------+
IDOCID   IDATE          RECTYPE     IRELEASE         IPAGES     FILLER
+-------------------------------------------------------------------------+
```

Record type C:

```
--------KEY--------
+-------------------------------------------------------------------------+
CRSEDOC   CDATE         RECTYPE     COURSENUM  LEVEL  CPAGES    FILLER
+-------------------------------------------------------------------------+
```

Without the ACCEPT attribute, each of the four record types must be described as separate segments in the Master File. A unique set of field names must be provided for record type M and record type I, although they have the same layout.

The generalized RECTYPE capability enables you to code just one set of field names that applies to the record layout for both record type M and I. The ACCEPT attribute can be used for any RECTYPE specification, even when there is only one acceptable value.

```
FILENAME=DOC2, SUFFIX=VSAM,$
SEGNAME=ROOT, SEGTYPE=SO,$
 GROUP=DOCNUM,     ALIAS=KEY,        A5,  A5,  $
  FIELD=DOCID,     ALIAS=SEQNUM,     A5,  A5,  $
 FIELD=FILLER,     ALIAS,=           A5,  A5,  $
 FIELD=RECTYPE,    ALIAS=DOCRECORD, A3,  A3,  ACCEPT = DN,$
 FIELD=TITLE,      ALIAS=,           A18, A18, $
SEGNAME=MANUALS, PARENT=ROOT, SEGTYPE=SO,$
 GROUP=MDOCNUM,    ALIAS=KEY, A10, A10,$
  FIELD=MDOCID,    ALIAS=MSEQNUM,    A5,  A5,  $
  FIELD=MDATE,     ALIAS=MPUBDATE,   A5,  A5,  $
 FIELD=RECTYPE,    ALIAS=MANUAL,     A3,  A3,  ACCEPT = M OR I,$
 FIELD=MRELEASE,   ALIAS=,           A7,  A7,  $
 FIELD=MPAGES,     ALIAS=,           I5,  A5,  $
 FIELD=FILLER,     ALIAS=,           A6,  A6,  $
SEGNAME=COURSES, PARENT=ROOT, SEGTYPE=SO,$
 GROUP=CRSEDOC,    ALIAS=KEY, A10, A10,$
  FIELD=CDOCID,    ALIAS=CSEQNUM,    A5,  A5,  $
  FIELD=CDATE,     ALIAS=CPUBDATE,   A5,  A5,  $
 FIELD=RECTYPE,    ALIAS=COURSE,     A3,  A3,  ACCEPT = C,$
 FIELD=COURSENUM, ALIAS=CNUM,       A4,  A4,  $
 FIELD=LEVEL,      ALIAS=,           A2,  A2,  $
 FIELD=CPAGES,     ALIAS=,           I5,  A5,  $
 FIELD=FILLER,     ALIAS=,           A7,  A7,  $
```

Combining Multiply-Occurring Fields and Multiple Record Types in VSAM

You can have two types of descendant segments in a single fixed-format, VSAM data source:

- Descendant segments consisting of multiply occurring fields.
- Additional descendant segments consisting of multiple record types.

Describing a Multiply Occurring Field and Multiple Record Types

In the data structure shown below, the first record, of type 01, contains several different sequences of repeating fields, all of which must be described as descendant segments with an OCCURS attribute. The data source also contains two separate records, of types 02 and 03, which contain information that is related to that in record type 01.

The relationship between the records of various types is expressed as parent-child relationships. The children that contain record types 02 and 03 do not have an OCCURS attribute. They are distinguished from their parent by the field declaration where field=RECTYPE.

01	T1	N1	B1	B2	C1	C1	C1	D1	D1	D1	D1	D1	D1	D1	B1	B2	C1	C1	D1	D1	D1	D1	D1	D1	D1

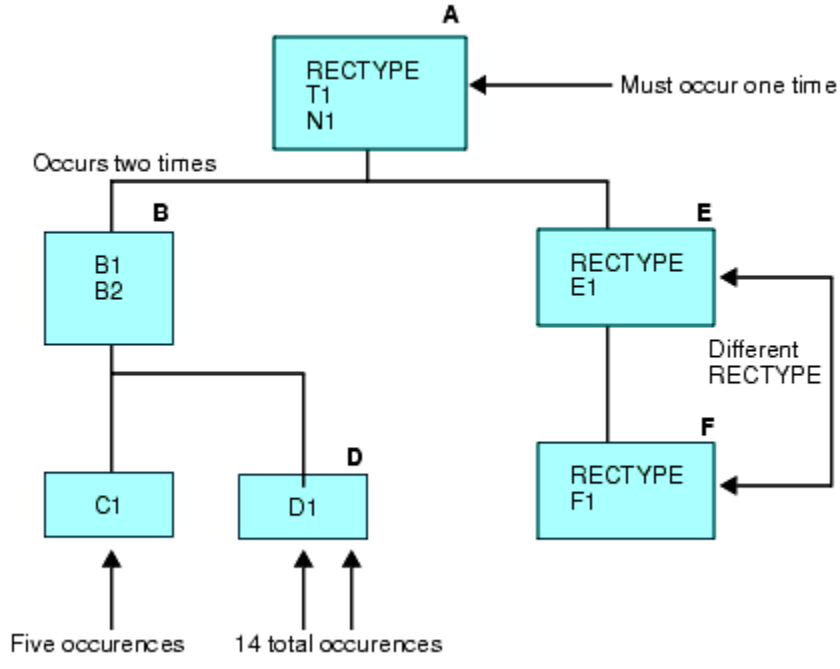
02	E1

03	F1

The description for this data source is:

```
FILENAME = EXAMPLE1, SUFFIX = FIX,$
 SEGNAME = A, SEGTYPE=S0,$
  FIELDNAME = RECTYPE   ,ALIAS = 01   ,USAGE = A2   ,ACTUAL = A2 ,$
  FIELDNAME = T1        ,ALIAS =      ,USAGE = A2   ,ACTUAL = A1 ,$
  FIELDNAME = N1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = B, PARENT = A, OCCURS = VARIABLE, SEGTYPE=S0,$
  FIELDNAME = B1        ,ALIAS =      ,USAGE = I2   ,ACTUAL = I2 ,$
  FIELDNAME = B2        ,ALIAS =      ,USAGE = I2   ,ACTUAL = I2 ,$
 SEGNAME = C, PARENT = B, OCCURS = B1, SEGTYPE=S0,$
  FIELDNAME = C1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = D, PARENT = B, OCCURS = 7,  SEGTYPE=S0,$
  FIELDNAME = D1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = E, PARENT = A, SEGTYPE=S0,$
  FIELDNAME = RECTYPE   ,ALIAS = 02   ,USAGE = A2   ,ACTUAL = A2 ,$
  FIELDNAME = E1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
 SEGNAME = F, PARENT = E, SEGTYPE=S0,$
  FIELDNAME = RECTYPE   ,ALIAS = 03   ,USAGE = A2   ,ACTUAL = A2 ,$
  FIELDNAME = F1        ,ALIAS =      ,USAGE = A1   ,ACTUAL = A1 ,$
```

It produces the following data structure:



Segments A, B, C, and D all belong to the same record type. Segments E and F each are stored as separate record types.

Note:

- Segments A, E, and F are different records that are related through their record types. The record type attribute consists of certain prescribed values, and is stored in a fixed location in the records. Records are expected to be retrieved in a given order. If the first record does not have a RECTYPE of 01, the record is considered to be a child without a parent. The next record can have a RECTYPE of either 01 (in which case, the first record is considered to have no descendants except the OCCURS descendants) or 02. A record with a RECTYPE of 03 can follow only a record with a RECTYPE of 02 (its parent) or another 03.
- The OCCURS descendants all belong to the record whose RECTYPE is 01. (This is not a necessary condition; records of any type can have OCCURS descendants.) Note that the OCCURS=VARIABLE segment, Segment B, is the rightmost segment within its own record type. If you look at the data structure, the pattern that makes up Segment B and its descendants (the repetition of fields B1, B2, C1, and D1) extends from the first mention of fields B1 and B2 to the end of the record.

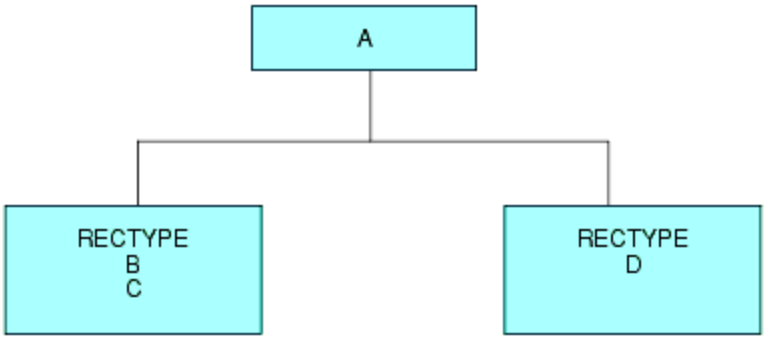
- Although fields C1 and D1 appear in separate segments, they are actually part of the repeating pattern that makes up the OCCURS=VARIABLE segment. Since they occur multiple times within Segment B, they are each assigned to their own descendant segment. The number of times field C1 occurs depends on the value of field B2. In the example, the first value of field B2 is 3; the second, 2. Field D1 occurs a fixed number of times, 7.

Describing a VSAM Repeating Group With RECTYPEs

Suppose you want to describe a data source that, schematically, looks like this:

```
A     RECTYPE  B C    RECTYPE   B C
A     RECTYPE  D      RECTYPE   D
```

You need to describe three segments in your Master File, with A as the root segment, and segments for B, C, and D as two descendant OCCURS segments for A:



Each of the two descendant OCCURS segments in this example depends on the RECTYPE indicator that appears for each occurrence.

All the rules of syntax for using RECTYPE fields and OCCURS segments also apply to RECTYPEs within OCCURS segments.

Since each OCCURS segment depends on the RECTYPE indicator for its evaluation, the RECTYPE must appear at the start of the OCCURS segment. This enables you to describe complex data sources, including those with nested and parallel repeating groups that depend on RECTYPEs.

Example: Describing a VSAM Repeating Group With RECTYPEs

In this example, B/C, and D represent a nested repeating group, and E represents a parallel repeating group.

A	RECTYPE B C	RECTYPE D	RECTYPE E	RECTYPE E

```
FILENAME=SAMPLE,SUFFIX=VSAM,$
 SEGNAME=ROOT,SEGTYPE=S0,$
  GROUP=GRPKEY      ,ALIAS=KEY ,USAGE=A8 ,ACTUAL=A8 ,$
   FIELD=FLD000     ,E00       ,A08      ,A08       ,$
   FIELD=A_DATA     ,E01       ,A02      ,A02       ,$
 SEGNAME=SEG001,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD=RECTYPE     ,A01       ,A01      ,ACCEPT=B OR C ,$
  FIELD=B_OR_C_DATA ,E02       ,A08      ,A08       ,$
 SEGNAME=SEG002,PARENT=SEG001,OCCURS=VARIABLE,SEGTYPE=S0,$
  FIELD=RECTYPE     ,D         ,A01      ,A01       ,$
  FIELD=D_DATA      ,E03       ,A07      ,A07       ,$
 SEGNAME=SEG003,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD=RECTYPE     ,E         ,A01      ,A01       ,$
  FIELD=E_DATA      ,E04       ,A06      ,A06       ,$
```

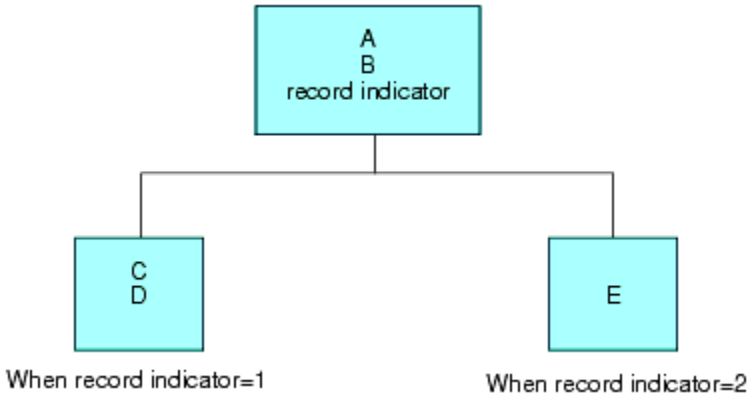
Describing a Repeating Group Using MAPFIELD

In another combination of record indicator and OCCURS, a record contains a record indicator that is followed by a repeating group. In this case, the record indicator is in the fixed portion of the record, not in each occurrence. Schematically, the record appears like this:

A B	record indicator (1)	C D	C D	C D
A B	record indicator (2)	E E		

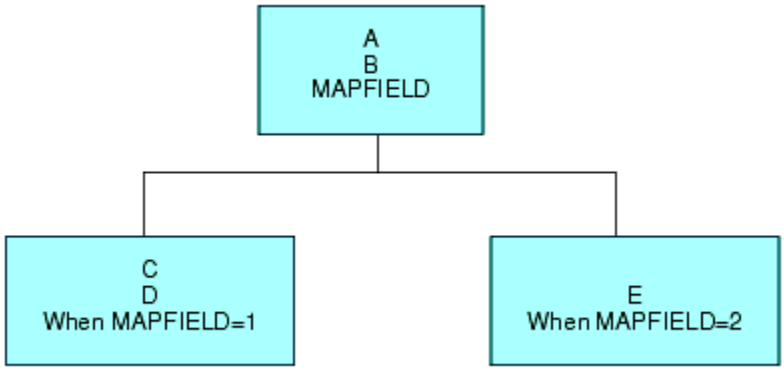
The first record contains header information, values for A and B, followed by an OCCURS segment of C and D that was identified by its preceding record indicator. The second record has a different record indicator and contains a different repeating group, this time for E.

The following diagram illustrates this relationship.



Since the OCCURS segments are identified by the record indicator rather than the parentA/B segment, you must use the keyword MAPFIELD. MAPFIELD identifies a field in the same way as RECTYPE, but since each OCCURS segments has its own value for MAPFIELD, the value of MAPFIELD is associated with each OCCURS segment by means of a complementary field named MAPVALUE.

The following diagram illustrates this relationship.



MAPFIELD is assigned as the ALIAS of the field that is the record indicator; it may have any name.

Syntax: How to Describe a Repeating Group With MAPFIELD

```
FIELD = name, ALIAS = MAPFIELD, USAGE = format, ACTUAL = format,$
```

where:

name

Is the name you choose to provide for this field.

ALIAS

MAPFIELD is assigned as the alias of the field that is the RECTYPE indicator.

USAGE

Follows the usual field format.

ACTUAL

Follows the usual field format.

The descendant segment values depend on the value of the MAPFIELD. They are described as separate segments, one for each possible value of MAPFIELD, and all descending from the segment that has the MAPFIELD. A special field, MAPVALUE, is described as the last field in these descendant segments after the ORDER field, if one has been used. The actual MAPFIELD value is supplied as the ALIAS of MAPVALUE.

Syntax: How to Use MAPFIELD for a Descendant Repeating Segment in a Repeating Group

```
FIELD = MAPVALUE, ALIAS = alias, USAGE = format, ACTUAL = format,
ACCEPT = {list|range} ,$
```

where:

MAPVALUE

Indicates that the segment depends on a MAPFIELD in its parent segment.

alias

Is the primary MAPFIELD value, if an ACCEPT list is not specified. If there is an ACCEPT list, this can be any value.

USAGE

Is the same format as the MAPFIELD format in the parent segment.

ACTUAL

Is the same format as the MAPFIELD format in the parent segment.

list

Is the list of one or more lines of specified MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the list is 255. Each item in the list must be separated by either a blank or the keyword OR. If the list contains embedded blanks or commas, it must be enclosed within single quotation marks ('). The list may contain a single MAPFIELD value.

For example:

```
FIELDNAME = MAPVALUE, ALIAS = A, USAGE = A1, ACTUAL = A1,
ACCEPT = A OR B OR C,$
```

range

Is a range of one or more lines of MAPFIELD values for records that have the same segment layout. The maximum number of characters allowed in the range is 255. If the range contains embedded blanks or commas, it must be enclosed in single quotation marks (').

To specify a range of values, include the lowest value, the keyword TO, and the highest value, in that order.

Example: Using MAPFIELD and MAPVALUE

Using the sample data source at the beginning of this section, the Master File for this data source looks like this:

```
FILENAME=EXAMPLE,SUFFIX=FIX,$
 SEGNAME=ROOT,SEGTYPE=S0,$
  FIELD =A,              ,A14   ,A14 ,$
  FIELD =B,              ,A10   ,A10 ,$
  FIELD =FLAG ,MAPFIELD  ,A01   ,A01 ,$
 SEGNAME=SEG001,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD =C,              ,A05   ,A05 ,$
  FIELD =D,              ,A07   ,A07 ,$
  FIELD =MAPVALUE ,1     ,A01   ,A01 ,$
 SEGNAME=SEG002,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0  ,$
  FIELD =E,              ,D12.2 ,D8  ,$
  FIELD =MAPVALUE ,2     ,A01   ,A01 ,$
```

Note: MAPFIELD can only exist on an OCCURS segment that has not been remapped. This means that the segment definition cannot contain POSITION=*fieldname*.

MAPFIELD and MAPVALUE may be used with SUFFIX=FIX and SUFFIX=VSAM data sources.

Establishing VSAM Data and Index Buffers

Two SET commands make it possible to establish DATA and INDEX buffers for processing VSAM data sources online.

The AMP sub-parameters BUFND and BUFNI enable users to enhance the I/O efficiency of TABLE, TABLEF, MODIFY, and JOIN against VSAM data sources by holding frequently used VSAM Control Intervals in memory, rather than on physical DASD. By reducing the number of physical Input/Output operations, you may improve job throughput. In general, BUFND (data buffers) increase the efficiency of physical sequential reads, whereas BUFNI (index buffers) are most beneficial in JOIN or KEYED access operations.

Syntax: How to Establish VSAM Data and Index Buffers

```
ENGINE VSAM SET BUFND {n|8}
ENGINE VSAM SET BUFNI {n|1}
```

where:

n

Is the number of data or index buffers. BUFND=8 and BUFNI=1 (eight data buffers and one index buffer) are the default values. The AMODE parameter determines whether the buffers are created above or below the line. SET AMODE=31 places the buffers above the line. SET AMODE=24 places the buffers below the line.

To determine how many buffers are in effect at any time, issue the query:

```
ENGINE VSAM SET ?
```

Using a VSAM Alternate Index

VSAM key-sequenced data sources support the use of alternate key indexes (keys). A key-sequenced VSAM data source consists of two components: an index component and a data component. The data component contains the actual data records, while the index component is the key used to locate the data records in the data source. Together, these components are referred to as the base cluster.

An alternate index is a separate, additional index structure that enables you to access records in a KSDS VSAM data source based on a key other than the data source primary key. For instance, you may usually use a personnel data source sequenced by Social Security number, but occasionally need to retrieve records sorted by job description. The job description field might be described as an alternate index. An alternate index must be related to the base cluster it describes by a path, which is stored in a separate data source.

The alternate index is a VSAM structure and is, consequently, created and maintained in the VSAM environment. It can, however, be described in your Master File, so that you can take advantage of the benefits of an alternate index.

The primary benefit of these indexes is improved efficiency. You can use it as an alternate, more efficient, retrieval sequence or take advantage of its potential indirectly, with screening tests (IF...LT, IF...LE, IF...GT, IF...GE, IF...EQ, IF...FROM...TO, IF...IS), which are translated into direct reads against the alternate index. You can also join data sources with the JOIN command through this alternate index.

It is not necessary to identify the indexed view explicitly in order to take advantage of the alternate index. An alternate index is automatically used when described in the Master File.

To take advantage of a specific alternate index during a TABLE request, provide a WHERE or IF test on the alternative index field that meets the above criteria. For example:

```
TABLE FILE CUST
PRINT SSN
WHERE LNAME EQ 'SMITH'
END
```

As you see in the Master File in *Describing a VSAM Alternate Index* on page 2566, the LNAME field is defined as an alternate index field. The records in the data source are retrieved according to their last names, and certain IF screens on the field LNAME result in direct reads. Note that if the alternate index field name is omitted, the primary key (if there is any) is used for a sequential or a direct read, and the alternate indexes are treated as regular fields.

Alternate indexes must be described in the Master File as fields with FIELDTYPE=I. The ALIAS of the alternate index field must be the file name allocated to the corresponding path name. Alternate indexes can be described as GROUPs if they consist of portions with dissimilar formats. Remember that ALIAS=KEY must be used to describe the primary key.

Only one record type can be referred to in the request when using alternate indexes, but there is no restriction on the number of OCCURS segments.

Note that the path name in the allocation is different from both the cluster name and the alternate index name.

If you are not sure of the path names and alternate indexes associated with a given base cluster, you can use the IDCAMS utility. (See the IBM manual entitled *Using VSAM Commands and Macros*, for details.)

Example: Describing a VSAM Alternate Index

Consider the following:

```
FILENAME = CUST, SUFFIX = VSAM,$
 SEGNAME = ROOT, SEGTYPE = S0,$
  GROUP = G, ALIAS = KEY, A10, A10,$
   FIELD = SSN,   SSN, A10, A10,$
   FIELD = FNAME, DD1, A10, A10, FIELDTYPE=I,$
   FIELD = LNAME, DD2, A10, A10, FIELDTYPE=I,$
```

In this example, SSN is a primary key and FNAME and LNAME are alternate indexes. The path data set must be allocated to the ddname specified in ALIAS= of your alternate index field. In this Master File, ALIAS=DD1 and ALIAS=DD2 each have an allocation pointing to the path data set. FNAME and LNAME must have INDEX=I or FIELDTYPE=I coded in the Master File. CUST must be allocated to the base cluster.

Example: Using IDCAMS

The following example demonstrates how to use IDCAMS to find the alternate index and path names associated with a base cluster named CUST.DATA:

First, find the alternate index names (AIX) associated with the given cluster.

```
IDCAMS input:
 LISTCAT CLUSTER ENTRIES(CUST.DATA) ALL

IDCAMS output (fragments):
 CLUSTER -------- CUST.DATA
  ASSOCIATIONS
   AIX ---------- CUST.INDEX1
   AIX ---------- CUST.INDEX2
```

This gives you the names of the alternate indexes (AIX): CUST.INDEX1 and CUST.INDEX2.

Next, find the path names associated with the given AIX name:

```
IDCAMS input:
 LISTCAT AIX ENTRIES (CUST.INDEX1 CUST.INDEX2) ALL

IDCAMS output (fragments):
 AIX ---------CUST.INDEX1
  ASSOCIATIONS
   CLUSTER -- CUST.DATA
   PATH ------CUST.PATH1
 AIX ---------CUST.INDEX2
  ASSOCIATIONS
   CLUSTER -- CUST.DATA
   PATH ------CUST.PATH2
```

This gives you the path names: CUST.PATH1 and CUST.PATH2.

This information, along with the TSO DDNAME command, may be used to ensure the proper allocation of your alternate index.

VSAM Record Selection Efficiencies

The most efficient way to retrieve selected records from a VSAM KSDS data source is by applying an IF screening test against the primary key. This results in a direct reading of the data using the data source index. Only those records that you request are retrieved from the file. The alternative method of retrieval, the sequential read, forces the adapter to retrieve all the records into storage.

Selection criteria that are based on the entire primary key, or on a subset of the primary key, cause direct reads using the index. A partial key is any contiguous part of the primary key beginning with the first byte.

IF selection tests performed against virtual fields can take advantage of these efficiencies as well, if the full or partial key is embedded in the virtual field.

The EQ and IS relations realize the greatest performance improvement over sequential reads. When testing on a partial key, equality logic is used to retrieve only the first segment instance of the screening value. To retrieve subsequent instances, NEXT logic is used.

Screening relations GE, FROM, FROM-TO, GT, EXCEEDS, IS-MORE-THAN, and NOT-FROM-TO all obtain some benefit from direct reads. The following example uses the index to find the record containing primary key value 66:

```
IF keyfield GE 66
```

It then continues to retrieve records by sequential processing, because VSAM stores records in ascending key sequence. The direct read is not attempted when the IF screening conditions NE, IS-NOT, CONTAINS, OMITS, LT, IS-LESS-THAN, LE, and NOT-FROM are used in the report request.

Reporting From Files With Alternate Indexes

Similar performance improvement is available for ESDS and KSDS files that use alternate indexes. An alternate index provides access to records in a key sequenced data set based on a key other than the primary key.

All benefits and limitations inherent with screening on the primary or partial key are applicable to screening on the alternate index or partial alternate index.

Note: It is not necessary to take an explicit indexed view to use the index.

Maintaining VSAM KSDS Data Sources

The Adapter for VSAM supports SQL update commands for VSAM KSDS data sources without the use of remote procedures. For example, UPDATE, INSERT, and DELETE will be translated into equivalent VSAM low-level read/write calls. No changes to the Master File are required in order to issue SQL INSERT, UPDATE, or DELETE commands against VSAM data sources.

General Guidelines for Maintaining VSAM KSDS Data Sources

The Structured Query Language (SQL) is intended to be used for access to relational tables, which are flat structures. A VSAM record, like a row in a relational table, has no hierarchical or network structure. However, unlike a relational table, a VSAM record may have repeating fields. It may also have multiple record types which require a different interpretation of the record layout from one record to another.

The adapter is designed to work against a single VSAM record. When a VSAM record can have multiple record layouts, the Master File describes the VSAM record as multiple segments, one for the fixed portion of the record (the root segment) and a separate descendant segment to describe each record layout. Similarly, when a VSAM record has repeating fields, the Master File describes this portion of the VSAM record as an OCCURS segment. However, to VSAM and SQL, these multiple segments still describe one VSAM record. Only one VSAM record can be the target of the SQL Data Manipulation Language (DML) statement. SQL set orientated behavior (multiple updates with one SQL statement) is not supported.

VSAM KSDS data sources have a unique key. To identify the record that is the target of the SQL statement, the SQL statement must supply the complete key for that record.

RECTYPEs in the Master File

When the Master File describes descendant segments with different RECTYPE values, you can insert a record without supplying a value for the RECTYPE field as long as the adapter can identify which RECTYPE is correct for the segment to be inserted. Since each RECTYPE is defined in a different segment in the Master File, if the field list for the INSERT contains a field name specific to that segment, the adapter can identify and insert the correct RECTYPE value.

OCCURS Segments in the Master File

When you want to update, insert, or delete a repeating group (OCCURS segment in the Master File), the technique you use depends on whether there are a fixed or variable number of occurrences defined in the Master File:

- For a fixed number of occurrences (OCCURS=*n* in the Master File), you must insert the record and then update a specific occurrence of the repeating group. The most efficient way to identify the occurrence is to specify a value for the ORDER field. You can also identify the occurrence by specifying any unique field value in the occurrence. The ORDER field is an internal counter maintained by the adapter for OCCURS segments.
- For a variable number of occurrences (OCCURS=*field* or OCCURS=VARIABLE in the Master File), you can insert a repeating group at the same time as the fixed portion of the record. Again, you identify the specific occurrence that is the target of the SQL command by supplying the ORDER field or a unique field value.

Note that when a fixed number of occurrences is specified in the Master File, any attempt to access an occurrence number greater than that fixed number generates an error.

VSAM records can be fixed or variable length. When you add or delete a repeating group in a fixed length record, the record length stays the same. When you add or delete a repeating group in a variable length record, the record length changes.

The WHERE clause of the SQL statement must contain the complete key for the target record. Any record referenced in the SQL query must, using the WHERE clause, be a unique occurrence of that record. If additional fields are included in the WHERE clause for any record, they will be used in conjunction with the key, to identify the segment.

Syntax: How to Issue an SQL INSERT Command in VSAM

```
INSERT INTO mfdname [(field1, field2 ...)] VALUES (value1, value2 ...)
```

General Rules

The following rules apply to SQL INSERT syntax:

- If you do not specify the field name list, the value list must contain values for all fields in all segments in the Master File as long as the Master File only specifies a single path VSAM file.
- If you specify the field name list, the value list only needs to specify, in field name list order, values for the field names supplied. As a minimum, you must supply all the key fields to the target record.

- If a record is not present, default values will be generated for all non-key fields not supplied.
- To insert a repeating field described with OCCURS=*n* in the Master File, insert the key field first and then use SQL UPDATE to enter a value in the repeating field. In the WHERE clause of the UPDATE command, specify a value for the ORDER field (or any unique field value) to identify which occurrence to update. Occurrences must be inserted in sequential order.

Syntax: How to Issue an SQL DELETE Command in VSAM

```
DELETE FROM mfdname WHERE fieldname=value [AND fieldname=value ...]
```

General Rules

The following rules apply to SQL DELETE syntax:

- To delete a complete record, you must supply the complete key for the target record. Only one target segment will be deleted.
- To delete a repeating field, you must specify the complete key for the record and a value for the ORDER field (or any unique field value) to identify which occurrence to delete.

Syntax: How to Issue an SQL UPDATE Command in VSAM

```
UPDATE mfdname  SET fieldname=value [SET fieldname=value...]
WHERE fieldname=value
[AND fieldname=value]
```

General Rules

The following rules apply to SQL UPDATE syntax:

- As a minimum, in the WHERE condition, you must supply all the key fields the record to be updated. Only one target record will be updated.
- To update a repeating field, you must specify the complete key for the record and a value for the ORDER field (or any unique field value) to identify which occurrence to update.

Syntax: How to Obtain the Number of Records Affected by an SQL INSERT, UPDATE, or DELETE Command

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Tip: You can change this setting manually or from the Web Console by clicking *Data Adapters* on the menu bar, clicking a configured adapter, and choosing *Change Settings* from the menu. The Change Settings pane opens.

```
*ENGINE* INT SET PASSRECS {ON|OFF}
```

where:

`INT`

Indicates that the PASSRECS setting in this command will be applied globally to all adapters that support SQL INSERT, UPDATE, and DELETE commands.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Reference: Processing Not Supported for Maintaining VSAM Data Sources

The following processing is not supported:

- Null data values.
- Embedded SELECT within INSERT.
- Cursor control, for example, WHERE CURRENT OF on UPDATE or DELETE processing.

Sample VSAM KSDS Master Files and Data Maintenance Examples

The following examples show how to insert, update, and delete records and occurrences of repeating groups in a VSAM KSDS data source.

Example: Inserting a VSAM KSDS Record With OCCURS=*n*

Consider the following Master File that describes a repeating field (FLD005 in segment R010) with OCCURS=3. Note that the Master File also describes the ORDER field, which is an internal counter for OCCURS segments:

```
FILE=VSMW0200          ,SUFFIX=VSAM,$
DATASET='EDAQA.VSMW0200.CLUSTER',   $
 SEGNAME=ROOT ,SEGTYPE=S0,$
 GROUP=GRPKEYA             ,ALIAS=KEY ,A8       ,A8      ,$
  FIELDNAME      =FLD000   ,          ,A8       ,A8      ,$
  FIELDNAME      =FLD001   ,          ,A2       ,A2      ,$
  FIELDNAME      =FLD003   ,          ,A4       ,A4      ,$
  FIELDNAME      =ACCOUNT  ,          ,A2       ,A2      ,$
 SEGNAME=R010 ,SEGTYPE=S0, PARENT=ROOT, OCCURS=3         ,$
  FIELDNAME      =FLD005   ,          ,A8       ,A8      ,$
  FIELDNAME      =ORDER    ,ORDER     ,I4       ,I4      ,$
```

The following SQL INSERT statement inserts a record with a key field of 1245678. An OCCURS segment cannot be inserted using SQL INSERT when the number of occurrences is fixed:

```
SQL
 INSERT INTO VSMW0200 (FLD000)
 VALUES('12345678');
END
```

The following SQL UPDATE statements place values in the occurrences of FLD005. The WHERE predicate in each SQL UPDATE command specifies the complete key for the record and a value for the ORDER field to identify which occurrence to update:

```
SQL
 UPDATE VSMW0200
 SET FLD005='00001001'
 WHERE FLD000='12345678' AND ORDER=1;
END
SQL
 UPDATE VSMW0200
 SET FLD005 = '00002002'
 WHERE FLD000 = '12345678' AND ORDER = 2;
END
SQL
 UPDATE VSMW0200
 SET FLD005 = '00003003'
 WHERE FLD000 = '12345678' AND ORDER = 3;
END
```

With OCCURS=*n*, the occurrence number referenced in the SQL statement must be less than or equal to *n*. If you reference an occurrence with a number greater than *n*, the occurrence is not updated and an error message is returned.

Example: Deleting an Occurrence of a Repeating Field in a VSAM KSDS Record

The following SQL DELETE statement deletes the first occurrence of the repeating field FLD005. The WHERE predicate identifies the complete key (FLD000='12345678') and the specific occurrence to delete (ORDER=1):

```
SQL
 DELETE FROM VSMW0200
 WHERE FLD000='12345678' AND ORDER=1;
END
```

Example: Deleting a VSAM KSDS Record

The following SQL DELETE statement deletes VSAM KSDS record whose key value is '12345678':

```
SQL
 DELETE FROM VSMW0200 WHERE FLD000='12345678';
END
```

Example: Inserting a VSAM KSDS Record With OCCURS=*field*

When a repeating field has a variable number of occurrences (OCCURS=*field* or OCCURS=VARIABLE), you can use an SQL INSERT statement to add a new record and insert data into a specific occurrence of the repeating field.

Consider the following Master File in which the number of occurrences of FLD005 is specified by the value in the ACCOUNT field:

```
FILE=VSMW0201          ,SUFFIX=VSAM,$
DATASET='EDAQA.VSMW0200.CLUSTER',   $
 SEGNAME=ROOT ,SEGTYPE=S0,$
 GROUP=GRPKEYA             ,ALIAS=KEY ,A8       ,A8      ,$
  FIELDNAME      =FLD000   ,           ,A8       ,A8      ,$
  FIELDNAME      =FLD001   ,           ,A2       ,A2      ,$
  FIELDNAME      =FLD003   ,           ,A4       ,A4      ,$
  FIELDNAME      =ACCOUNT   ,           ,I2       ,I2      ,$
 SEGNAME=R010 ,SEGTYPE=S0,  PARENT=ROOT, OCCURS=ACCOUNT  ,$
  FIELDNAME      =FLD005   ,           ,A8       ,A8      ,$
  FIELDNAME      =ORDER    ,ORDER     ,I4       ,I4      ,$
```

The following SQL INSERT statement inserts a new record with key FLD000='12345678' and the value '00001001' in the first occurrence of FLD005. The ORDER field identifies the occurrence that is inserted, but it represents an internal counter and does not exist in the VSAM record:

```
SQL
 INSERT INTO VSMW0201 (FLD000,FLD005,ORDER)
 VALUES('12345678','00001001',1) ;
END
```

To add a second occurrence of FLD005, issue an SQL INSERT statement for key FLD000='12345678' and the value 2 for the ORDER field. Note that the occurrences must be added in order:

```
SQL
 INSERT INTO VSMW0201 (FLD000,FLD005,ORDER)
 VALUES('12345678','00002002',2) ;
END
```

If you attempt to add an occurrence that would make the record length exceed the maximum record length defined for the VSAM data source, the occurrence is not added and an error message is returned.

Example: Inserting a Record in a VSAM KSDS Record With Record Types

Consider the following Master File that describes two OCCURS segments with a variable number of occurrences. The key field is FLD000, which is followed by FLD001. Next comes any combination of the two possible layouts for the variable portion of the record. Each repeating group starts with a value that indicates the type of layout that follows. If the RECTYPE field contains the value 2, the repeating group is an instance of segment SEG002, and it contains one 8-byte field (FLD002). If the RECTYPE field contains the value 3, the repeating group is an instance of segment SEG003, and it contains two 8-byte fields (FLD003 and FLD004):

```
FILE=VSMW0208,          SUFFIX=VSAM,$
DATASET='EDAQA.VSMW0200.CLUSTER',   $
 SEGNAME=ROOT,SEGTYPE=S0,$
  GROUP =GRPKEY      ,ALIAS=KEY   ,USAGE=A8,ACTUAL=A8,$
   FIELD=FLD000      ,E00         ,A08     ,A08      ,$
  FIELD =FLD001      ,E01         ,A02     ,A02      ,$
 SEGNAME=SEG002,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0,$
  FIELD =RECTYPE     ,2           ,A01     ,A01      ,$
  FIELD =FLD002      ,E02         ,A08     ,A08      ,$
 SEGNAME=SEG003,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0,$
  FIELD =RECTYPE     ,3           ,A01     ,A01      ,$
  FIELD =FLD003      ,E03         ,A08     ,A08      ,$
  FIELD =FLD004      ,E04         ,A08     ,A08      ,$
```

The following SQL INSERT statement adds an occurrence of SEG002. The INSERT statement references FLD002, which identifies the INSERT as an instance of SEG002. The adapter automatically inserts the value 2 in the RECTYPE field, even though it is not specified as part of the insert statement:

```
SQL
 INSERT INTO VSMW0208 (FLD000,FLD002)
 VALUES('12345678','ANTIGUA') ;
END
```

The following SQL INSERT statement adds an occurrence of SEG003. The INSERT statement references fields FLD003 and FLD004, which identifies the INSERT as an instance of SEG003. The adapter automatically inserts the value 3 in the RECTYPE field, even though it is not specified as part of the insert statement:

```
SQL
 INSERT INTO VSMW0208 (FLD000,FLD003,FLD004)
 VALUES('12345678','BRAZIL','AMERICA') ;
END
```

Example: **Updating an Occurrence of a Repeating Group in a VSAM KSDS Record With Record Types**

The following SQL UPDATE statement updates an occurrence of FLD004 in the record with key '12345678'. The ORDER field is not defined in this Master File. The specific occurrence of FLD004 to update is identified by specifying a unique value of FLD003:

```
SQL
 UPDATE VSMW0208
 SET FLD004='S AMER'
 WHERE FLD000='12345678' AND FLD003='BRAZIL';
END
```

Example: **Deleting an Occurrence of a Repeating Group in a VSAM KSDS Record With Record Types**

The following SQL DELETE statement deletes an occurrence of FLD002 in the record with key '12345678'. The ORDER field is not defined in this Master File. The specific occurrence of FLD002 to delete is identified by specifying its unique value ('ANTIGUA'):

```
SQL
 DELETE FROM VSMW0208
 WHERE FLD000='12345678' AND FLD002='ANTIGUA';
END
```

Example: **Inserting a Record in a VSAM KSDS Record With a MAPFIELD**

Consider the following Master File that describes two OCCURS segments with a variable number of occurrences. The key field is a concatenation of fields FLD000 and FLAG (the MAPFIELD). If the FLAG field contains the value 2, the remainder of the record contains a variable number of occurrences of segment SEG002, which consists of one 8-byte field (FLD002). If the flag field contains the value 3, the remainder of the record consists of a variable number of occurrences of segment SEG003, which consists of two 8-byte fields (FLD003 and FLD004):

```
FILE=VSMW0209,          SUFFIX=VSAM,$
DATASET='EDAQA.VSMW0200.CLUSTER',   $
 SEGNAME=ROOT,SEGTYPE=S0,$
  GROUP =GRPKEY      ,ALIAS=KEY   ,USAGE=A8,ACTUAL=A8,$
   FIELD=FLD000      ,E00         ,A07     ,A07      ,$
   FIELD=FLAG        ,MAPFIELD    ,A01     ,A01      ,$
 SEGNAME=SEG002,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0,$
  FIELD =FLD002      ,E02         ,A08     ,A08      ,$
  FIELD =MAPVALUE    ,2           ,A01     ,A01      ,$
 SEGNAME=SEG003,PARENT=ROOT,OCCURS=VARIABLE,SEGTYPE=S0,$
  FIELD =FLD003      ,E03         ,A08     ,A08      ,$
  FIELD =FLD004      ,E04         ,A08     ,A08      ,$
  FIELD =MAPVALUE    ,3           ,A01     ,A01      ,$
```

The following SQL INSERT statement adds an occurrence of SEG002. The INSERT statement must provide a value for the FLAG field because it is part of the key:

```
SQL
 INSERT INTO VSMW0209 (FLD000,FLAG,FLD002)
 VALUES('1234567','2','ANTIGUA') ;
END
```

The following SQL INSERT statement adds an occurrence of SEG003:

```
SQL
 INSERT INTO VSMW0209 (FLD000,FLAG,FLD003,FLD004)
 VALUES('1234567','3','BRAZIL','AMERICA') ;
END
```

Example: Updating an Occurrence of a Repeating Field in a VSAM KSDS Record With a MAPFIELD

The following SQL UPDATE statement updates an occurrence of FLD004 in the record with key '12345673'. The ORDER field is not defined in this Master File. The specific occurrence of FLD004 to update is identified by specifying a unique value of FLD003:

```
SQL
 UPDATE VSMW0209
 SET FLD004='S AMER'
 WHERE GRPKEY='12345673' AND FLD003='BRAZIL';
END
```

Example: Deleting an Occurrence of a Repeating Field in a VSAM KSDS Record With a MAPFIELD

The following SQL DELETE statement deletes an occurrence of FLD002 in the record with key '12345672'. The ORDER field is not defined in this Master File. The specific occurrence of FLD002 to delete is identified by specifying its unique value ('ANTIGUA'):

```
SQL
 DELETE FROM VSMW0209
 WHERE GRPKEY='12345672' AND FLD002='ANTIGUA';
END
```

Using VSAM Relative Record Data Set (RRDS) Files

The Adapter for VSAM supports both direct and sequential retrieval of fixed- and variable-length records from RRDS files, as well as modification of those records. RRDS files are VSAM files that are accessed through Relative Record Number keys (RRNs) and are processed in basically the same manner as key-sequenced (KSDS or KSEQ) VSAM files.

The RRN field that serves as the primary (unique) key for the segment follows all real fields in the physical root segment of the Master File and establishes the sequence for the records. The RRN field description must contain the following attributes:

```
ALIAS=RRN, USAGE=I10, ACTUAL=I4
```

Records in RRDS files are processed sequentially if the request does not provide a screening condition or join for the RRN-associated field.

Syntax: How to Specify a Relative Record Number Field in a Master File

```
FILE=RRDSVSAM, SUFFIX=VSAM, DATASET=PGMYAN.RRDSVSAM.CLUSTER, $
SEGNAME=ROOT ,SEGTYPE=S0 ,$
FIELD=CODE , ,USAGE=A2 ,ACTUAL=A2 ,$
FIELD=NAME , ,USAGE=A10 ,ACTUAL=A10 ,$
FIELD=VALUE , ,USAGE=P2 ,ACTUAL=Z1 ,$
FIELD=ACCESS_KEY ,ALIAS=RRN ,USAGE=I10 ,ACTUAL=I4 ,$
```

Example: Applying Unique Keys With Records in RRDS Files

The following examples show how to view and apply unique Relative Record Number (RRN) key values when working with records in RRDS files.

Printing the records with their key values

```
TABLE FILE RRDSVSAM
PRINT CODE NAME VALUE ACCESS_KEY
END
```

Screening records by unique key values

```
TABLE FILE RRDSVSAM
PRINT CODE NAME VALUE ACCESS_KEY
IF ACCESS_KEY  GE 5
IF ACCESS_KEY  LE 15
END
```

Retrieving records by unique key values

```
TABLE FILE RRDSVSAM
PRINT CODE NAME VALUE ACCESS_KEY
IF ACCESS_KEY  EQ 5 OR 7 OR  9
END
```

Modifying records by unique key values

```
MODIFY FILE RRDSVSAM ECHO
FREEFORM ACCESS_KEY
MATCH ACCESS_KEY
ON NOMATCH COMPUTE CODE = 'A1';
ON NOMATCH COMPUTE NAME = 'RECORD 20';
ON NOMATCH INCLUDE
ON MATCH REJECT
DATA
ACCESS_KEY=20
END
```

Updating matching records in RRDS files by unique key values

```
MODIFY FILE RRDSVSAM ECHO
FREEFORM ACCESS_KEY
MATCH ACCESS_KEY
ON NOMATCH REJECT
ON MATCH
TYPE "CODE: <D.CODE>, NAME: <D.NAME>, VALUE: <D.VALUE>";
ON MATCH COMPUTE VALUE = 9;
ON MATCH UPDATE VALUE
DATA
ACCESS_KEY=20
END
```

Deleting RRDS records based on unique key values

```
MODIFY FILE RRDSVSAM ECHO
FREEFORM ACCESS_KEY
MATCH ACCESS_KEY
ON NOMATCH REJECT
ON MATCH
TYPE "CODE: <D.CODE>, NAME: <D.NAME>, VALUE: <D.VALUE>";
ON MATCH DELETE
DATA
ACCESS_KEY=7
END
```

Reviewing SQL Updates to VSAM Data Sources

The Adapter for VSAM supports SQL INSERT, UPDATE, and DELETE operations against VSAM data sources for use in accessing and maintaining legacy systems. This facilitates development and implementation of new applications based on VSAM data sources.

Reference: Obtain Counts of Rows Updated or Deleted

While processing SQL transactions, you can issue the PASSRECS command to obtain counts of rows affected by each successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax: How to Count Records Updated, Inserted or Deleted

```
*ENGINE* INT SET PASSRECS {ON|OFF}
```

where:

`INT`

Indicates that the PASSRECS setting in this command will be applied globally to all adapters that support SQL INSERT, UPDATE, and DELETE commands.

`ON`

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

`OFF`

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Chapter 106 Using the Adapter for WAND Sentiment Analysis

This section describes how to configure the WAND Sentiment Analysis Adapter.

In this chapter:

WAND Sentiment Analysis Adapter Overview

The WAND Sentiment Analysis Adapter is used to score structured and unstructured textual content by identifying positive, neutral, and negative sentiment found within emails, documents, and database records. Textual data from a data source can be passed to the adapter by:

- Joining the column containing the textual data from the data source to the column within the WAND Sentiment Analysis Adapter used to define the textual data to be scored.
- A report which uses Cluster Join metadata. The Cluster Join metadata already contains the join from the column containing the textual data from the data source to the column within the WAND Sentiment Analysis Adapter used to define the textual data to be scored.
- Using a WHERE/IF condition to pass textual data directly to the column within the WAND Sentiment Analysis Adapter used to define the textual data to be scored.

Understanding the Scoring System for WAND Sentiment Analysis

The score returned from the WAND Sentiment Analysis Adapter ranges from -1 to 1.

- A score of -1 identifies the sentiment of the textual data that was passed to the adapter as extremely negative.
- A score of 0 identifies the sentiment of the textual data that was passed to the adapter as neutral.

- A score of 1 identifies the sentiment of the textual data that was passed to the adapter as tremendously positive.

Installing, Configuring, and Updating the WAND Taxonomy Server

This section describes how to install, configure, and update the WAND Taxonomy Server.

Procedure: How to Install the WAND Taxonomy Server

The WAND Taxonomy Server requires a Microsoft Windows 2008 Server environment or higher.

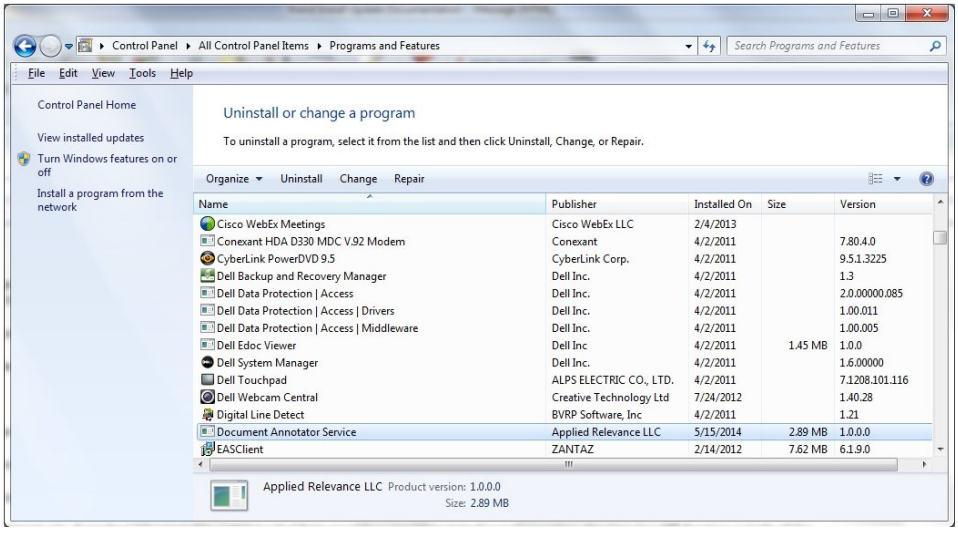
1. If a previous version of the WAND Taxonomy Server exists, then perform the following steps to uninstall the software:

   a. Create a backup of the Sentiment Taxonomy file (Sentiment.artx), which is located in the following directory:

   ```
   C:\Program Files (x86)\Applied Relevance\DocumentAnnotatorService
   \repository
   ```

   Save this backup copy to a different directory outside of the C:\Program Files (x86)\Applied Relevance directory structure (for example, C:\temp).

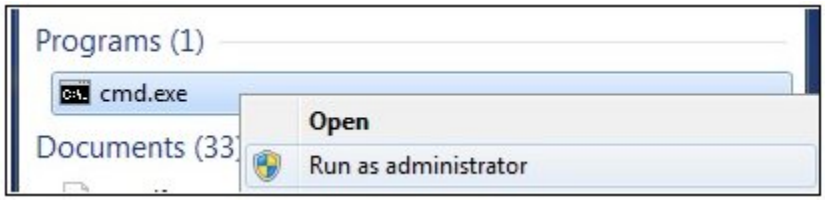
   b. Uninstall *Document Annotator Service* from the Control Panel, as shown in the following image.



   c. Delete the following directory:

   ```
   C:\Program Files (x86)\Applied Relevance\DocumentAnnotatorService
   ```

2. Obtain the WAND Taxonomy Server software.
3. If necessary, unzip the installation software to a temporary directory on your file system (for example, C:\Wand).
4. Search for the cmd.exe file from the Start menu and run it as an Administrator, as shown in the following image.

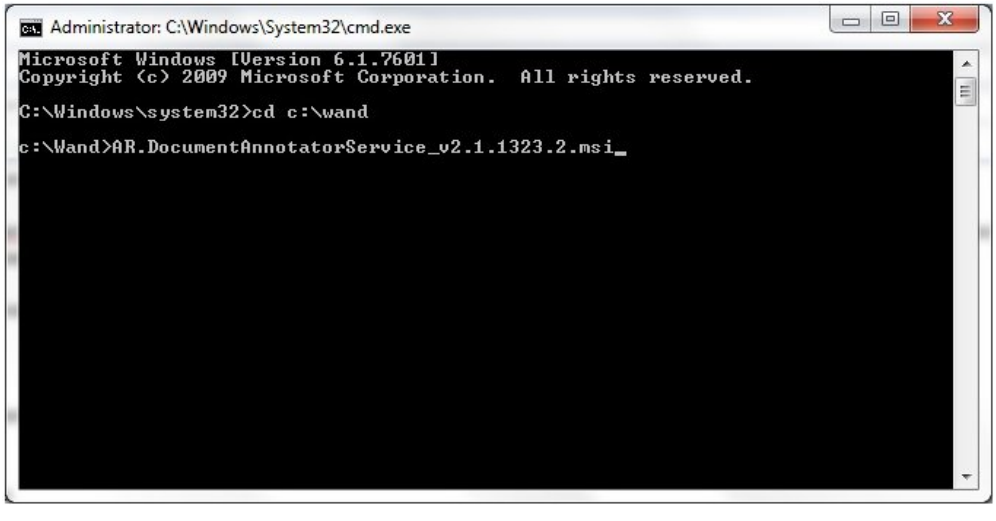


5. Navigate to the directory that contains the Wand Taxonomy Server installation software.

   Type the .msi file name for the Wand Taxonomy Server installation.

   For example:

   ```
   AR.DocumentAnnotatorService_v2.1.1323.2.msi
   ```

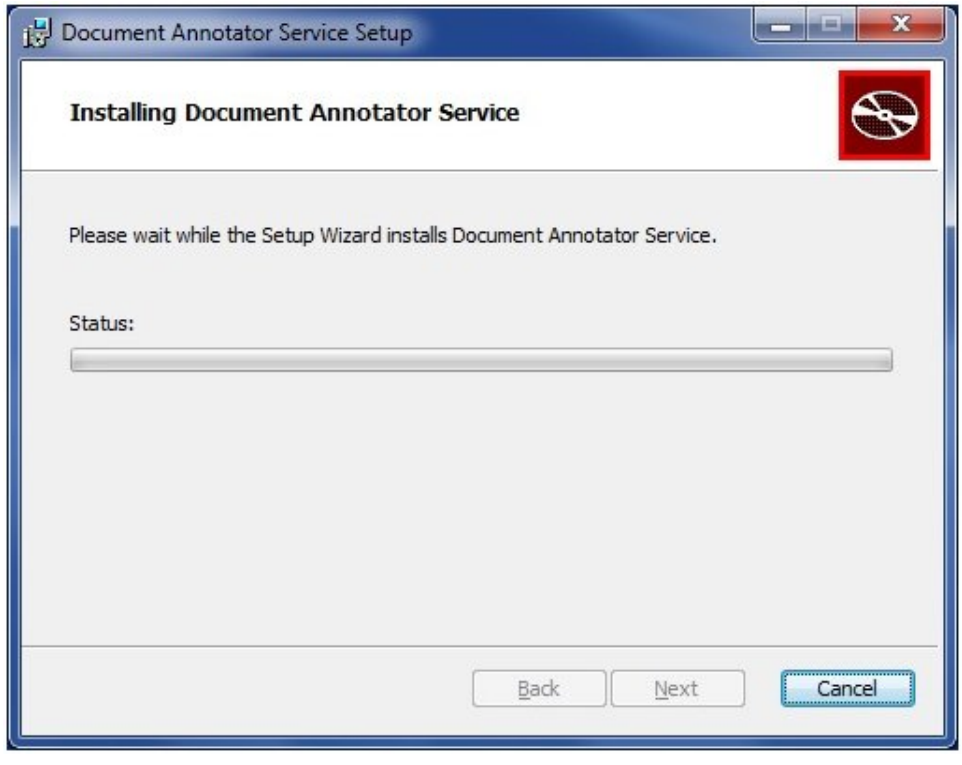


6. Press Enter.

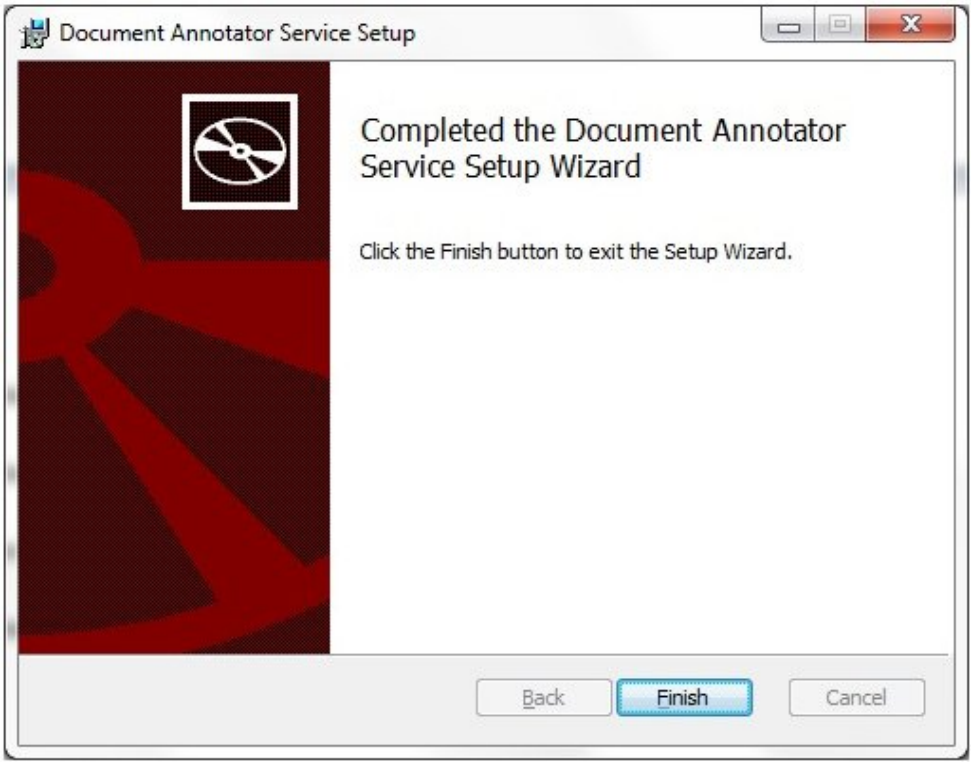
The Document Annotator Service Setup dialog box opens, as shown in the following image.

7. Select *I accept the terms in the License Agreement* and then click *Install*.

The Setup Wizard installs the Document Annotator Service, as shown in the following image.



8. When the installation of the Document Annotator Service is complete, click *Finish*, as shown in the following image.



Note: If the version of the WAND Taxonomy Server software installed is an update to a previous version, then follow the instructions in *How to Update the WAND Sentiment Taxonomy* on page 2589. You must update the installed Sentiment taxonomy with the Sentiment taxonomy that was backed up in Step 1.

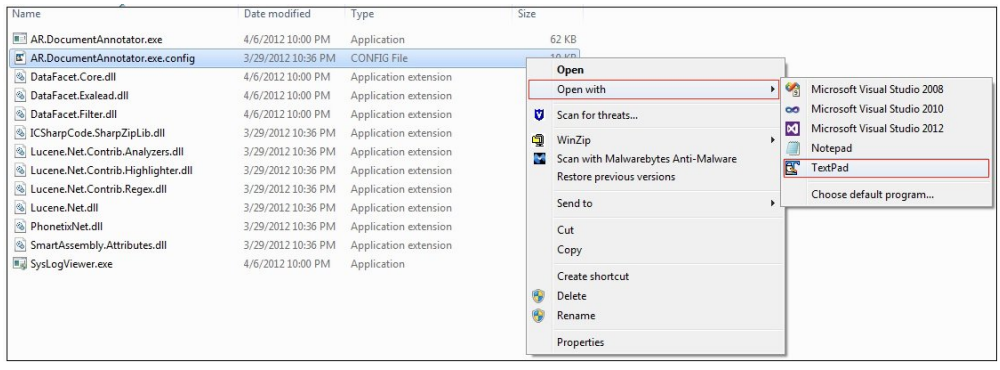
Procedure: How to Configure the WAND Taxonomy Server

The WAND Taxonomy Server must be configured so that the host name for the Taxonomy Server installation is either the machine name or IP address.

1. Navigate to the following directory on your file system:

```
C:\Program Files (x86)\Applied Relevance\DocumentAnnotatorService\bin
```

2. Edit the *AR.DocumentAnnotator.exe.config* file using a text editor.



The following section in this configuration file contains the address for the WAND Taxonomy Server:

```
<host>
  <baseAddresses>
    <add baseAddress="http://localhost:4701" />
  </baseAddresses>
</host>
```

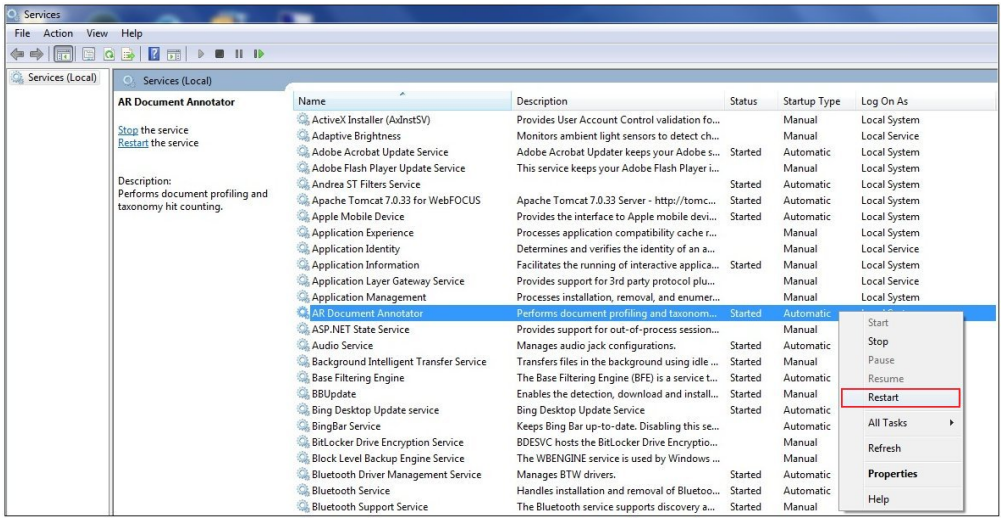
3. If required, modify the value in the <baseAddress> element so that the host name is the machine name or IP address where the WAND Taxonomy Server is installed.

   For example:

```
<host>
  <baseAddresses>
    <add baseAddress="http://wandserver.ibi.com:4701" />
  </baseAddresses>
</host>
```

4. Save the changes made in the *AR.DocumentAnnotator.exe.config* file.

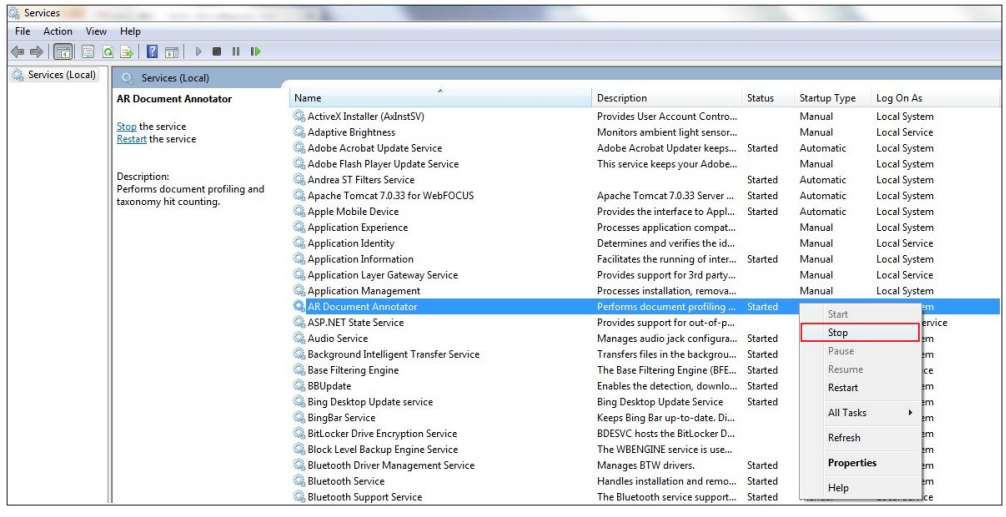
5. Open the Services utility on Windows through the Control Panel and *restart* the AR Document Annotator service, as shown in the following image.



Procedure: How to Update the WAND Sentiment Taxonomy

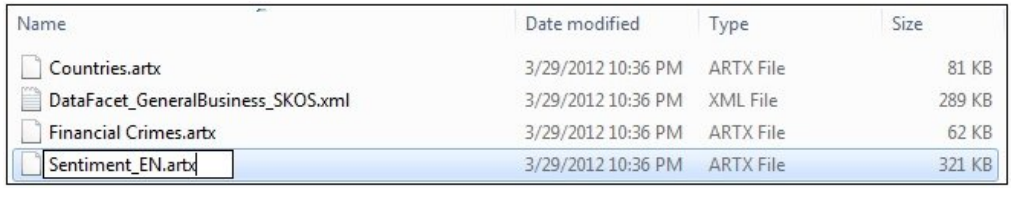
The Sentiment.artx file, which is located in the *C:\Program Files (x86)\Applied Relevance \DocumentAnnotatorService\repository* directory contains the Sentiment Taxonomy used for scoring textual data. As a Sentiment Taxonomy update or a localized Sentiment Taxonomy in a different language is made available, the Sentiment.artx file must be replaced with the updated version.

1. Open the Services utility on Windows through the Control Panel and *stop* the AR Document Annotator service, as shown in the following image.



2. Rename the Sentiment.artx file, which is located in the following directory:

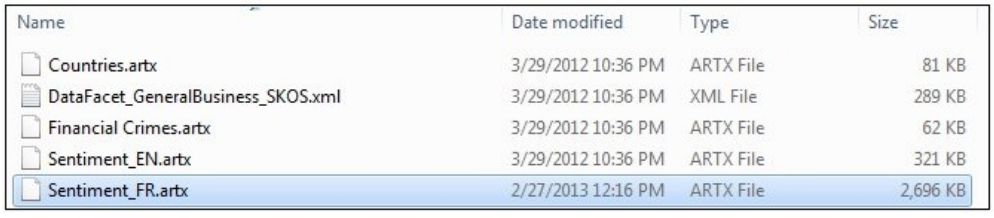
```
C:\Program Files (x86)\Applied Relevance\DocumentAnnotatorService
\repository
```



3. Copy the new Sentiment Taxonomy to the following directory:

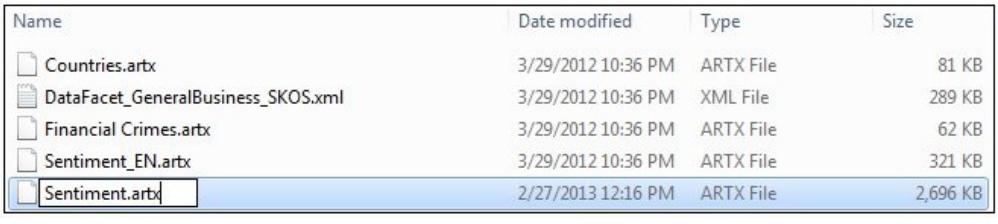
```
C:\Program Files (x86)\Applied Relevance\DocumentAnnotatorService
\repository
```

The following example shows the installation of a French Sentiment Taxonomy.



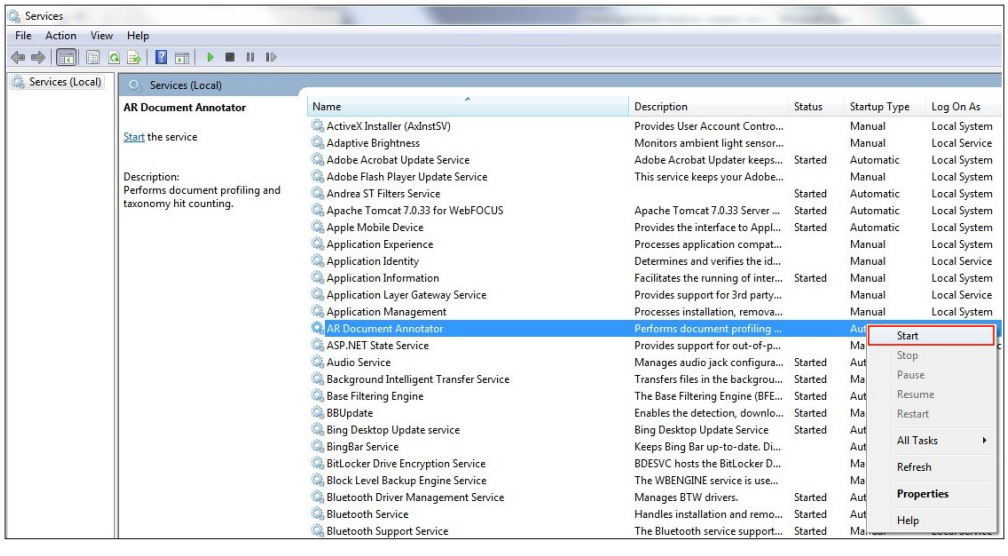
Name	Date modified	Type	Size
Countries.artx	3/29/2012 10:36 PM	ARTX File	81 KB
DataFacet_GeneralBusiness_SKOS.xml	3/29/2012 10:36 PM	XML File	289 KB
Financial Crimes.artx	3/29/2012 10:36 PM	ARTX File	62 KB
Sentiment_EN.artx	3/29/2012 10:36 PM	ARTX File	321 KB
Sentiment_FR.artx	2/27/2013 12:16 PM	ARTX File	2,696 KB

4. Rename the updated Sentiment Taxonomy so that the file name is Sentiment.artx, as shown in the following image.



Name	Date modified	Type	Size
Countries.artx	3/29/2012 10:36 PM	ARTX File	81 KB
DataFacet_GeneralBusiness_SKOS.xml	3/29/2012 10:36 PM	XML File	289 KB
Financial Crimes.artx	3/29/2012 10:36 PM	ARTX File	62 KB
Sentiment_EN.artx	3/29/2012 10:36 PM	ARTX File	321 KB
Sentiment.artx	2/27/2013 12:16 PM	ARTX File	2,696 KB

5. Open the Services utility on Windows through the Control Panel and *start* the AR Document Annotator service, as shown in the following image.



Installing and Using the WAND Taxonomy Editor

After the Taxonomy Server is installed and the Sentiment Taxonomy (Sentiment.artx file) within the *C:\Program Files (x86)\Applied Relevance\DocumentAnnotatorService\repository* directory is updated to the latest version, there might be a need to score words that are not present in the taxonomy. Therefore, these words would have to be added to the Sentiment Taxonomy and assigned to one of the following categories:

- Neutral
- Satisfied
- Very Satisfied
- Very Dissatisfied
- Dissatisfied

The WAND Taxonomy Editor is used to add and categorize words to the Sentiment Taxonomy.

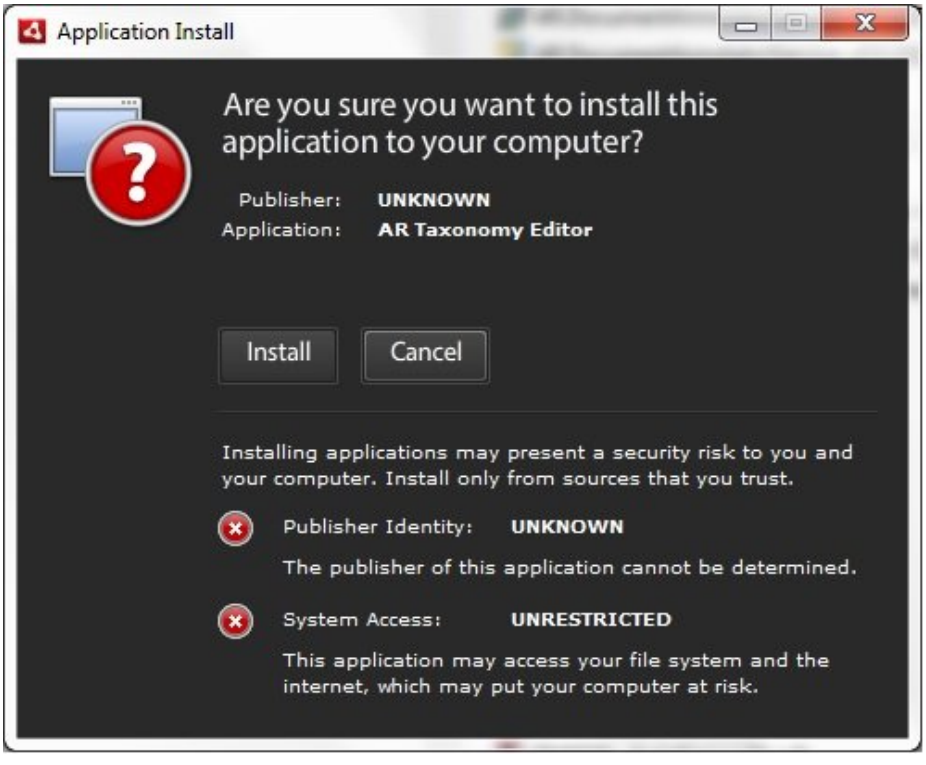
Procedure: How to Install the WAND Taxonomy Editor

1. Obtain the WAND Taxonomy Editor software.
2. Ensure that Adobe Air is installed on your system.

   Adobe Air can be downloaded from:

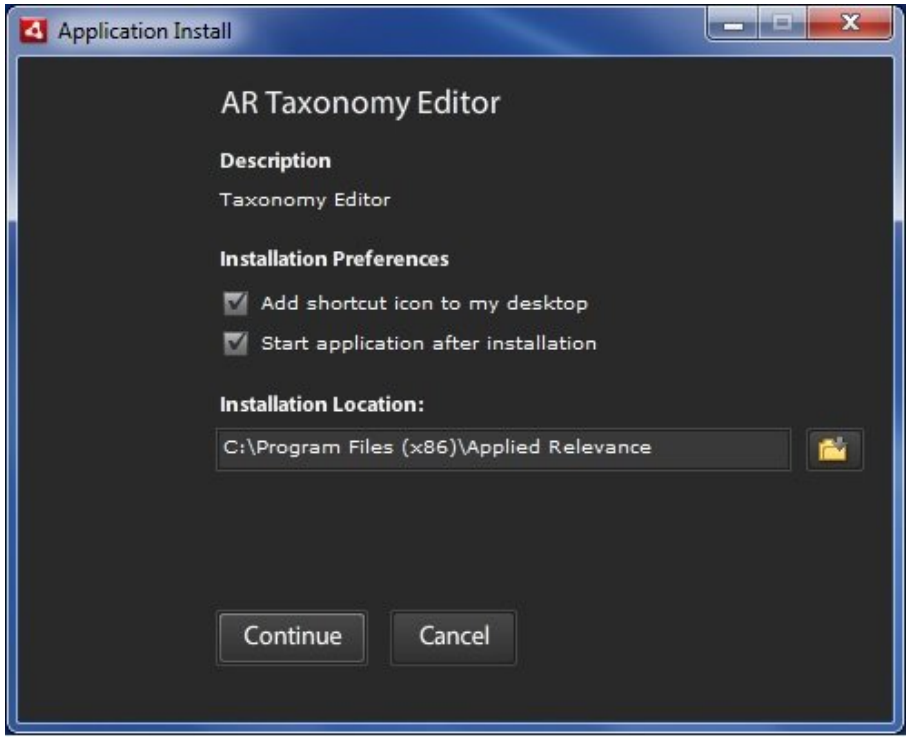
   http://get.adobe.com/air/
3. Double-click the *ARTaxonomyEditor_v2.1.1318.0.air* file.

The Application Install dialog box opens, as shown in the following image.



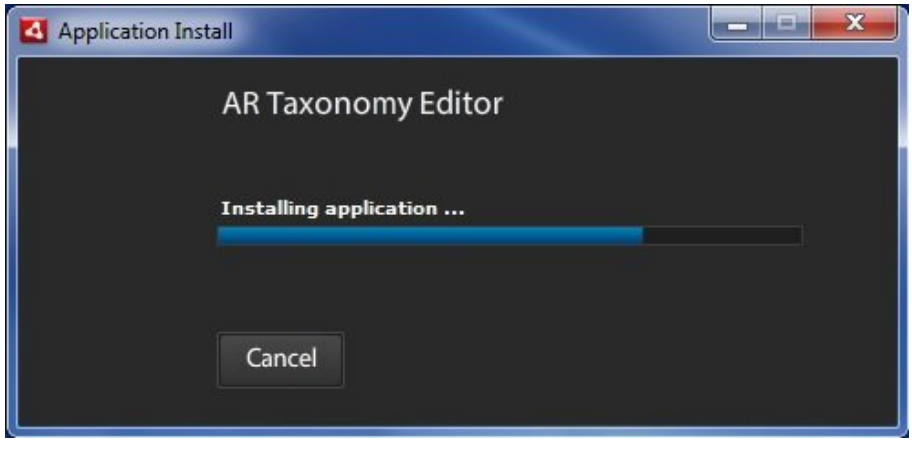
4. Click *Install*.

The AR Taxonomy Editor installation pane is displayed, as shown in the following image.

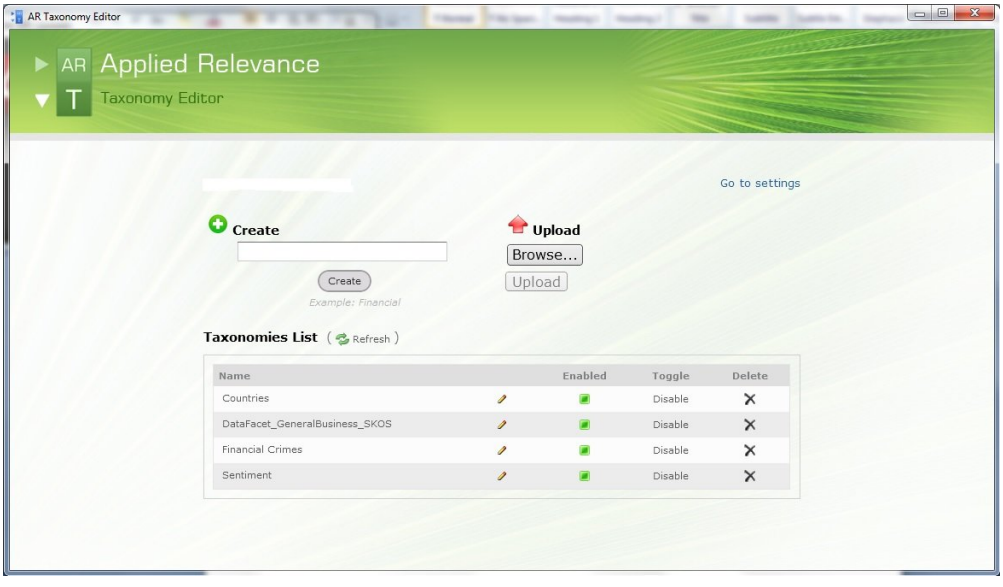


5. Click *Continue*.

   During the installation, a progress bar is displayed, as shown in the following image.

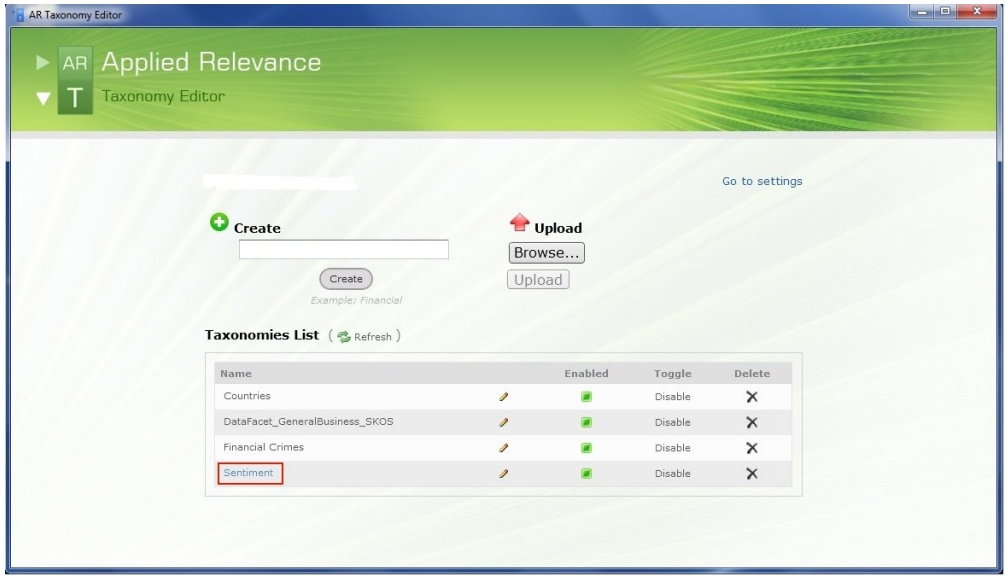


After the installation is complete, the AR Taxonomy Editor is displayed, as shown in the following image.

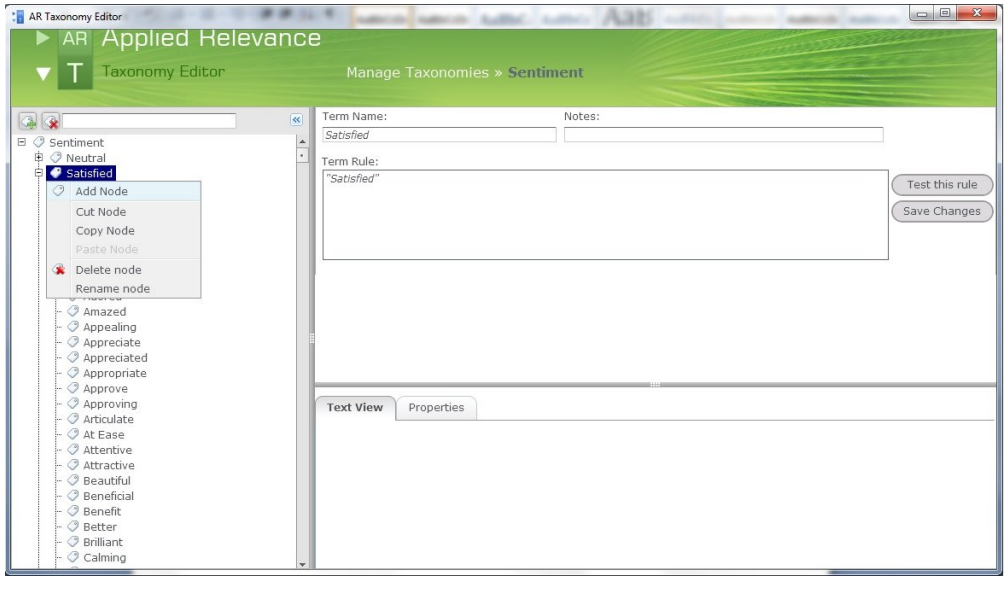


Procedure: How to Use the WAND Taxonomy Editor

1. In the AR Taxonomy Editor, click *Sentiment* from the Taxonomies List, as shown in the following image.



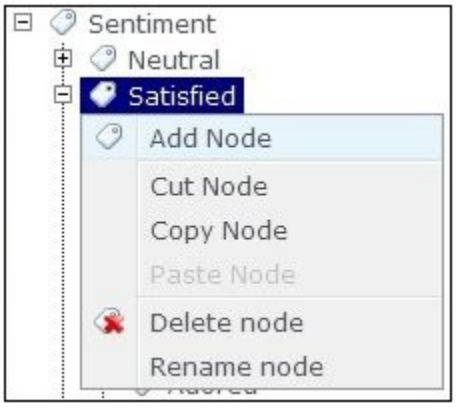
The Sentiment window of the AR Taxonomy Editor is displayed, as shown in the following image.



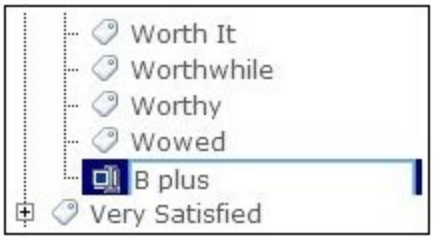
The Sentiment Taxonomy categories include Neutral, Satisfied, Very Satisfied, Very Dissatisfied, and Dissatisfied.

To categorize words, a node must be added under the category that will reflect the appropriate scoring.

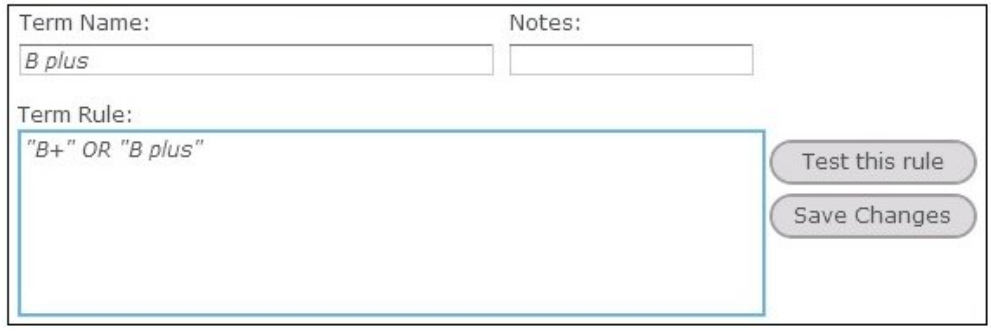
2. Right-click on a category in the left pane (for example, Satisfied) and click *Add Node* from the context menu, as shown in the following image.



3. In the new field that is added in the left pane, enter a name for the new node (for example, B plus), as shown in the following image.



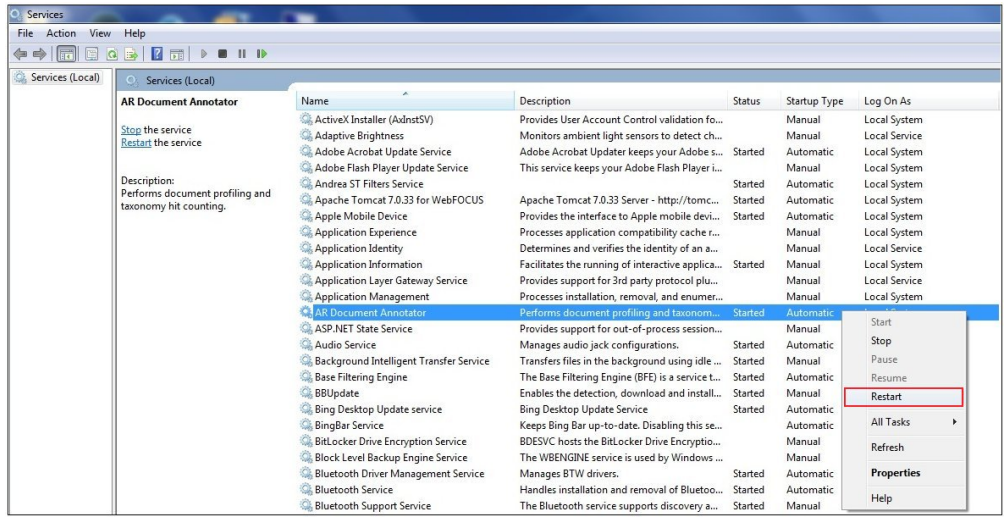
4. In the right pane, enter a new term rule for the node in the Term Rule field, as shown in the following image.



   For example, entering *"B+" OR "B plus"* will be scored as Satisfied.

5. Click *Save Changes* to save the new addition to the Sentiment Taxonomy.

6. To apply the changes, open the Services utility on Windows through the Control Panel and *restart* the AR Document Annotator service, as shown in the following image.



Configuring the WAND Sentiment Analysis Adapter

The WAND Sentiment Analysis Adapter is a part of the Social Media group of adapters that are managed by the WebFOCUS Reporting Server.

Procedure: How to Configure the WAND Sentiment Analysis Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

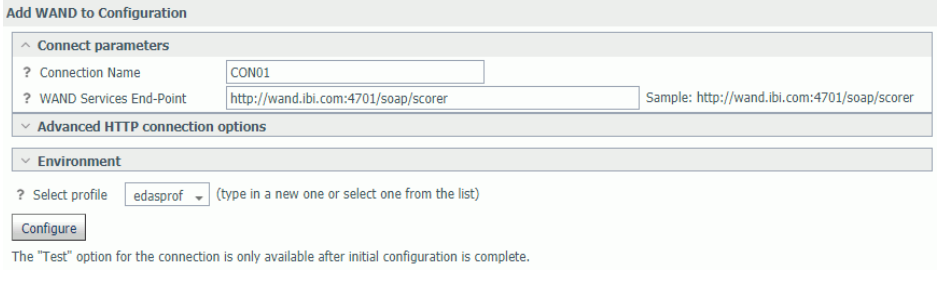
   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the *WAND* node and select *Configure*.

   The Add WAND to Configuration pane opens, as shown in the following image.



5. Enter a name to identify the connection (for example, WAND) in the Connection Name field.

   The format of the WAND Services End-Point URL is:

   ```
   http://host:4701/soap/scorer
   ```

   where:

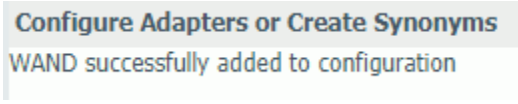
   host

   Is the machine name or IP address where the Taxonomy Server is installed. For example:

   ```
   http://wandserver.ibi.com:4701/soap/scorer
   ```

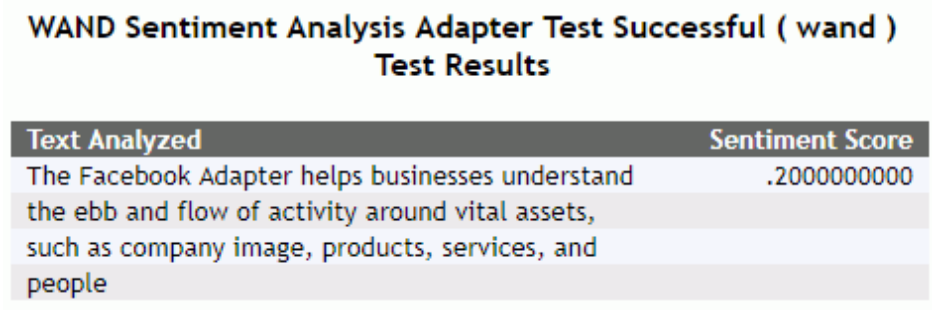
6. Click *Configure*.

The Configure Adapters or Create Synonyms pane opens, as shown in the following image.



7. Click *Test* to ensure that the WAND Sentiment Analysis Adapter is configured properly.

   The Testing Wand connection pane opens and displays the test results, as shown in the following image.



Reference: Connection Attributes for WAND Sentiment Analysis

The following list describes the connection attributes for the WAND Sentiment Analysis Adapter.

Connection Name

Logical name used to identify this particular set of connection attributes. The default is CON01.

WAND Services End-Point

The URL that is used to connect to the WAND Sentiment Analysis environment. For example:

```
http://wand.ibi.com:4701/soap/scorer
```

PROXY Server IP Address

IP address of the proxy server. For example:

```
170.115.249.42
```

PROXY Port

Port number on which the proxy server listens. The default port number is 80.

Select profile

Select a profile from the drop-down list to indicate the level of profile in which to store the connection attributes. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (user.prf) or a group profile if available on your platform (using the appropriate naming convention), choose New Profile from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

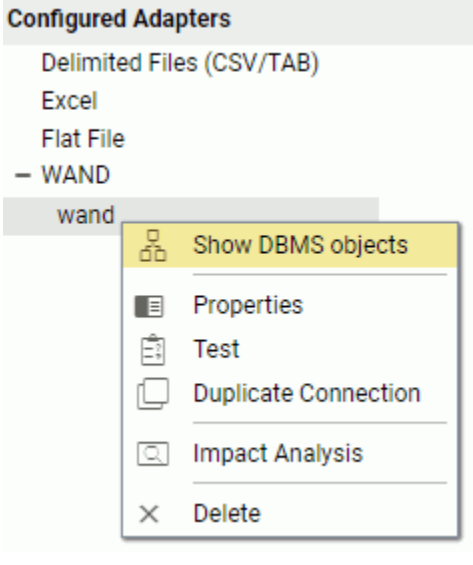
Store the connection attributes in the server profile (edasprof).

Creating Metadata and Sample Reports for the WAND Sentiment Analysis Adapter

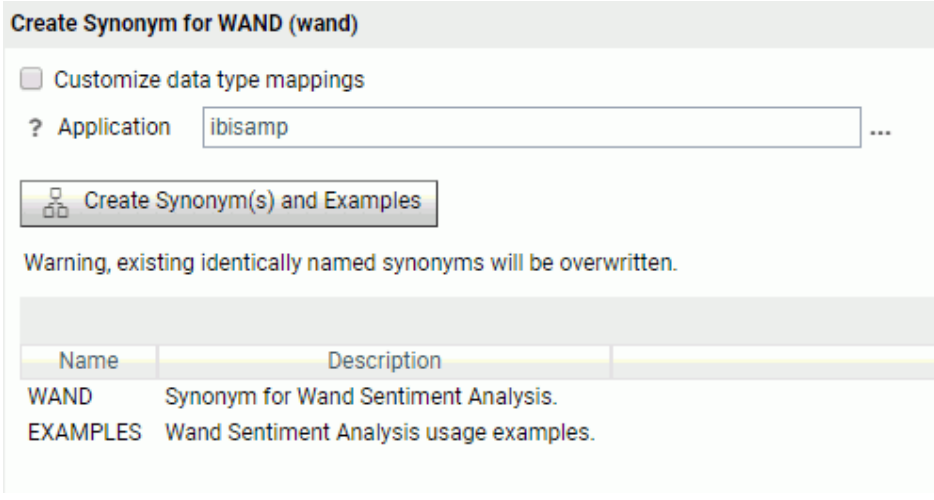
Create Synonym for the WAND Adapter creates the metadata used by reports to perform Sentiment Analysis scoring as well as sample reports which utilize the metadata.

Procedure: How to Create Metadata and Sample Reports

1. From the WebFOCUS Reporting Server Web Console or the Data Management Console, expand the *Adapters* folder, *Configured* folder, and then the *WAND* folder.
2. Right-click the configured connection for the WAND Sentiment Analysis Adapter (for example, wand) and select *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for WAND pane opens, as shown in the following image.

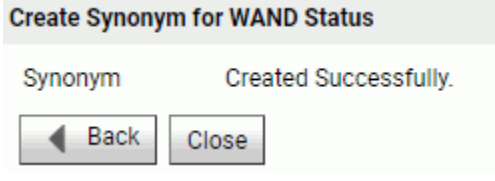


3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata is to be stored.

   The sample reports are stored within the *wandsampl* subdirectory of the application.

4. Click *Create Synonym*.

   The Create Synonym for WAND Status pane opens and indicates that the synonym was created successfully, as shown in the following image.



WAND Sentiment Analysis Adapter Examples

This section describes the metadata and sample reports for the WAND Sentiment Analysis Adapter.

Reference: WAND Sentiment Analysis Adapter Metadata

The following table lists and describes the available metadata for the WAND Sentiment Analysis Adapter.

Metadata	Description

wandscore	Metadata is used for interacting with the WAND web service for sentiment analysis scoring.
	
	The TEXT field would contain the textual data that is to be analyzed and scored by the WAND Taxonomy Server. Textual data can be joined from a column within a table to the TEXT field or set within a WHERE/IF condition.
	
	The following example uses a JOIN statement:
	
	`JOIN DOCLINE IN wand/wandsampl/ wand_sample_fix TO TEXT IN wand/wandscore END`
	
	The following example uses a WHERE/IF condition:
	
	`WHERE (TEXT CONTAINS 'The Facebook Adapter helps businesses')`
	
	A sentiment score between -1 and 1 is returned within the SCORERESULT field.

wandsampl/wand_sample_fix	Metadata that defines the sample text file (wandsampl/wand_sample.txt) used for the *wand_sample_join* and *wand_sample_cluster* sample reports.
wandsampl/wand_sample_cluster	Cluster join between *wandsampl/wand_sample_fix* and *wandscore*.

Reference: **WAND Sentiment Analysis Adapter Sample Reports**

The following table lists and describes the sample reports for the WAND Sentiment Analysis Adapter.

Sample Report	Description
wandsampl/wand_sample_join	Scores the text passed from the *wandsampl/wand_sample.txt* file performed through a JOIN statement.
wandsampl/wand_sample_cluster	Scores the text passed from the *wandsampl/wand_sample.txt* file using the Cluster Join master *wandsampl/wand_sample_cluster*.
wandsampl/wand_sample_where	Scores the text passed within a WHERE statement.

Chapter 107

Using the Adapter for Web Services

The Adapter for Web Services allows client applications to access Web Services providers. The adapter converts:

- Internal data manipulation language (DML) requests or SQL requests into Web Services requests.
- Web Services responses into answer sets.

In this chapter:

Configuring the Adapter for Web Services

Configuring the adapter consists of specifying connection and authentication information for at least one connection.

Declaring Connection Attributes

In order to access the Web Services provider hosting the target Web service, the adapter requires connection information. You supply this information using the SET CONNECTION_ATTRIBUTES command. You can:

- Enter connection and authentication information in the Web Console or the Data Management Console configuration panes. The consoles add the command to the profile you select: the global server profile (edasprof.prf), a user profile (*user.prf*), or a group profile (if supported on your platform).
- Manually add the command in the global server profile (edasprof.prf), in a user profile (*user.prf)*, or in a group profile (if supported on your platform).

You can declare connections to more than one Web Services provider by including multiple SET CONNECTION_ATTRIBUTES commands.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.
5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Reference: Connection Attributes for Web Services

The Web Services adapter is under the *Procedures* group folder.

The following list describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Connection name

Logical name used to identify this particular set of connection attributes. The default is CON01.

WSDL URL

URL of the WSDL that describes the Web service. Addresses can begin with:

file://

http://

https://

This value is required for creating a synonym only; otherwise, it is ignored.

PROXY Server IP Address

IP address of the proxy server, which intercepts requests and forwards them to the real server.

Enable HTTPS

Select one of the following options for secure HTTP connections.

- **None.** Does not enable HTTPS. This is the default value.
- **Windows.** Enables Windows HTTPS connections.
- **SSL.** Enables Secure Socket Layer HTTPS connections. If you select this option, additional SSL fields open that you need to configure.

PROXY Port

Port number on which the proxy server listens. The default port number is 80.

KERBEROS SPN

When Security is set to Trusted, this is the Service Principal Name used by the client to uniquely identify an instance of a service. For example

```
mydaemon/foo:4761
```

SSL Certificate

Location of locally-stored user-provided server x.509 certificates for SSL authentication. The certificate file is used to authenticate the server to which the adapter is connecting.

SSL Mutual Authentication

When checked, mutual authentication is enabled.

SSL Certificate Type

Is the type of SSL certificate being used. Options are:

- **Trusted.** For *trusted*, the trusted certificate file (*trustedcertfile*) points to a file of CA certificates in PEM format, as illustrated below:

  ```
  -----BEGIN CERTIFICATE-----
  ... (CA certificate in base64 encoding) ...
  -----END CERTIFICATE-----
  ```

  A trusted certificate file can contain several CA certificates. You can add text before, between, and after certificates. For example, to provide descriptions of certificates.

- **non-trusted.** Requires you to configure a key file, pass phrase, and label.

SSL Certificate Key file

Is the private key used for creating the client X.509 certificate in PEM format. This option is used together with a certificate for a non-trusted connection.

SSL Certificate Pass phrase

Is the password used to unlock the key file. The value is needed only if the key file is encrypted.

SSL Certificate Label

Identifies a certificate in the file if the file contains more than one certificate.

Security

There are three methods by which a user can be authenticated when connecting to a Web Services provider. The default value is *None*, which does not authenticatE users:

- **Explicit.** The user ID and password are explicitly specified for each connection and passed to Web Services, at connection time, for authentication.
- **Password Passthru.** The user ID and password received from the client application are passed to Web Services, at connection time, for authentication.
- **Trusted.** The adapter connects to the database using the database rules for an impersonated process that are relevant to the current operating system.

User

Appears when Security is Explicit. Primary authorization ID by which you are known to Web Services.

Password

Appears when Security is Explicit. Password associated with the primary authorization ID.

Chained Authentication

Click to enable chained authentication. Chained execution of an authentication operation is used when the web server returns a response containing a cookie or a token, to be used in the subsequent processing operation.

Synonym Application

This option is used only for chained authentication. For details about this process, see *How to Configure the Adapter for Chained Execution* on page 2612.

Select profile

Select a profile from the drop-down menu to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.

If you wish to create a new profile, either a user profile (*user*.prf) or a group profile if available on your platform (using the appropriate naming convention), choose *New Profile* from the drop-down menu and enter a name in the Profile Name field (the extension is added automatically).

Store the connection attributes in the server profile (edasprof).

End Points

This option appears on a second configuration pane, after you click the *Next* button.

Select the End Points URL for the Web Services provider from the drop-down list.

Syntax: How to Declare Connection Attributes Manually

The user ID and password are explicitly specified for each connection and passed to Web Services, at connection time, for authentication. The syntax is

```
ENGINE SOAP SET CONNECTION_ATTRIBUTES connection/
[user[,pswd]]:'endpoint_URL WSDL_URL [AUTH:applid/mf]
[fieldname="value" ...]
[PROXYS:proxy_IP_address] [PROXYP:{80|port_number}]
[CERTIFICATE:cert_location [cert_key=key_file][cert_phrase=pass_phrase]
[cert_label= cert_label]]'
```

where:

SOAP

Indicates the adapter.

connection

Is the logical name used to identify this particular set of attributes.

user,*pswd*

If specified, the user ID and password are passed to the authentication operation defined in the AUTH parameter at connection time.

For chained authentication, these are *required* fields if defined in the authentication Master File, either in the header or in the description of the group. For related information, see *How to Configure the Adapter for Chained Execution* on page 2612.

endpoint_URL

Is the URL of the Web Services provider.

WSDL_URL

Is the URL of the WSDL that describes the Web service. This is used to create a synonym.

applid/mf

For chained authentication, this is the location on the server of the Master File that describes the associated authentication operation. For related information, see *How to Configure the Adapter for Chained Execution* on page 2612.

`fieldname=` *value*

For chained authentication in which you are passing additional parameters to an authentication operation, *fieldname* corresponds to the field defined with NEED_VALUE *value* in the ACCESS_PROPERTY parameter in the Master File, except for fields using the reserved names USERNAME and PASSWORD.

For an illustration, see *Passing Additional Parameters to an Authentication Operation* on page 2615.

Note:

- Values provided in the SET CONNECTION_ATTRIBUTES command overwrite default values set in XDEFAULT parameter in the Master File.
- Field values must be enclosed in double quotation marks (") and cannot contain double or single quotation marks.

proxys_IP_address

Is the IP address of the proxy server, which intercepts requests and forwards them to the real server.

port_number

Is the port number on which the proxy server listens. The default port number is 80.

cert_location

Is the location of locally-stored user-provided server x.509 certificates (Trusted CA) for SSL authentication. The certificate file is used to authenticate the server to which the adapter is connecting.

The trusted certificate file *(trustedcertfile)* points to a file of CA certificates in PEM format, as illustrated below:

```
-----BEGIN CERTIFICATE-----
... (CA certificate in base64 encoding) ...
-----END CERTIFICATE-----
```

A trusted certificate file can contain several CA certificates. You can add text before, between, and after certificates. For example, to provide descriptions of certificates.

If you wish to specify a Non-trusted certificate, enter the following additional information:

```
[cert_key=key_file] [cert_phrase=pass_phrase] [cert_label=cert_label]
```

key_file

Is the private key used for creating the client X.509 certificate in PEM format. This entry is required for a non-trusted connection.

pass_phrase

Is the password used to unlock the key file. This entry is needed only if the key file is encrypted.

cert_label

Identifies a certificate in the file if the file contains more than one certificate.

Example: Declaring Connection Attributes

The following SET CONNECTION_ATTRIBUTES command allow the application to access the Web application named SAMPLEAPP.

```
ENGINE SOAP SET CONNECTION_ATTRIBUTES esriR/45678,4456789DFAFC:
 'http://arcweb.esri.com/services/v2/AddressFinder
 file://D:\users\WSDL\ESRI\AddressFinder.wsdl AUTH:ESRI/gettoken'
```

Security for Web Services

Chained execution of an authentication operation is used when the web server returns a response containing a cookie or a token, to be used in the subsequent processing operation. Chained execution is defined in the connection string associated with the requested operation. When a cookie is used, both connection strings (authentication and execution) must contain the same end point URL.

The Adapter for Web Services supports the following security facilities:

- **Cookies**, containing user credentials, which are in effect for the length of a TSCOM agent session (that is, between user connect and user disconnect). The scope of the cookie is the single TSCOM agent. The received cookie is sent back to the senders IP, along with subsequent SOAP requests.
- **Identification token** values, which are returned by an authentication operation and are acceptable to associated execution operations.

Chained authentication supports the explicit and passthru (with PING capabilities) security models.

To use chained authentication you must declare the connection for authentication, then create a Master File that contains the authentication specifications (using an authentication string) and store it an application. You can then use the authentication metadata to create one or more associated execution connection strings. For details, see *How to Configure the Adapter for Chained Execution* on page 2612.

Note that when the connection string contains user ID and password information and chained authentication is not defined, basic HTTP authentication is used. That is, the user ID and password are encrypted using the x64 algorithm and then used to establish the HTTP connection.

Procedure: How to Configure the Adapter for Chained Execution

Follow these steps to configure the adapter for chained authentication:

1. Create a connection string using a WSDL URL that describes authentication operations. This connection is needed to specify the authentication end-point URL.

   To complete this step, in the Web Console or the Data Management Console Connection Parameters pane, type the *WSDL URL*, click *Next*, and select an *End Points* URL from the drop-down list.

   Keep in mind that if a cookie (rather than a token) is to be used in the subsequent processing operation, you must specify the *same* end point URL for this authentication connection string and for the execution connection string that you specify in step 3.

2. Create synonyms for relevant authentication operations.

   To complete this step, from the Web Console or the Data Management Console, create the synonym, then open the graphical Synonym Editor to edit the synonym using the following guidelines.

   Authentication synonyms *must* include the following input fields:

   `USERNAME`

   The user ID value is taken from the connection string defining the operation to be executed.

   `PASSWORD`

   The password value is taken from the connection string defining the operation to be executed.

   Authentication synonyms *may* also include fields with the following access properties.

The following fields must belong to segments describing the SOAP request:

`ACCESS_PROPERTY=NEED_VALUE`

Defines fields that provide additional parameters for an authentication operation. You can supply default values in XDEFAULT parameters. Values provided in the associated connection string overwrite the default values.

There may be more than one such field.

The following fields must belong to segments describing the SOAP response:

`ACCESS_PROPERTY=AUTHRESP`

Defines fields that describe the result of an authentication operation. Correct response values must be provided in the ACCEPT attribute (using the OR predicate if more than one value is acceptable).

There may be more than one such field. The operation is considered invalid if at least one of the fields contains a non-acceptable value.

`ACCESS_PROPERTY=AUTHTOKEN`

Defines a field that contains a response token to be passed as an input value to the operation to be executed. There can be only one such field. If none is defined, the authentication operation is expected to return a cookie.

3. Create a connection string using a WSDL URL that describes the operation and the end point of the execution request, and a previously created authentication operation synonym.

   To complete this step, from the Web Console or the Data Management Console Connection Parameters pane, type the *WSDL URL*, then from the *Via MFD/Select Applications* drop-down list, select the location on the server of the Master File that describes the associated authentication operation. Click *Next* and select the *End Points* URL from the drop-down list.

4. Create synonyms for relevant execution operations.

   From the Web Console or the Data Management Console, create the synonym.

5. Ensure that the operation synonyms have an input field that describes the authenticating token (if needed) using ACCESS_PROPERTY=AUTHTOKEN.

   To complete this step, from the Web Console or the Data Management Console, open the graphical Synonym Editor to edit the synonym as described.

For related information, see *Connection Attributes for Web Services* on page 2606 and *Creating Synonyms* on page 2616.

Example: Configuring Chained Authentication

Connection strings:

```
ENGINE SOAP SET CONNECTION_ATTRIBUTES esriA/,:
'https://arcweb.esri.com/services/v2/Authentication
file://D:\users\WSDL\ESRI\Authentication.wsdl'

ENGINE SOAP SET CONNECTION_ATTRIBUTES esriR/109618,8366C1649D9DFAFC:
'http://arcweb.esri.com/services/v2/AddressFinder
file://D:\users\WSDL\ESRI\AddressFinder.wsdl AUTH:ESRI/gettoken'
```

Master File fragment describing an authentication operation:

```
FILENAME=GETTOKEN, SUFFIX=SOAP , $
  SEGMENT=GETTOKEN, SEGTYPE=S0, $
  FIELDNAME=USERNAME, ALIAS=username, USAGE=A30, ACTUAL=A30,
    ACCESS_PROPERTY=(NEED_VALUE), $
  FIELDNAME=PASSWORD, ALIAS=password, USAGE=A30, ACTUAL=A30,
    ACCESS_PROPERTY=(NEED_VALUE), $
  FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
    ACCESS_PROPERTY=(INTERNAL),$
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=GETTOKEN,
    POSITION=__RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=getToken0Out, USAGE=A1, ACTUAL=A1,
    ACCESS_PROPERTY=(INTERNAL), $
  FIELDNAME=RESULT, ALIAS=Result, USAGE=A120, ACTUAL=A120,
    ACCESS_PROPERTY=(AUTHTOKEN),
  REFERENCE=RESPONSE, PROPERTY=ELEMENT, $
```

Master File fragment describing a data retrieval operation:

```
FILENAME=FINDADDRESS, SUFFIX=SOAP , $
SEGMENT=FINDADDRESS, SEGTYPE=S0, $
GROUP=ADDRESS, ALIAS=address, USAGE=A210, ACTUAL=A210, $
 FIELDNAME=HOUSENUMBER, ALIAS=houseNumber, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='2815', $
 FIELDNAME=STREET, ALIAS=street, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='PRAIRIE AVE.', $
 FIELDNAME=INTERSECTION, ALIAS=intersection, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
 FIELDNAME=CITY, ALIAS=city, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='MIAMI BEACH', $
 FIELDNAME=STATE_PROV, ALIAS=state_prov, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='FL', $
 FIELDNAME=ZONE, ALIAS=zone, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='33140', $
 FIELDNAME=COUNTRY, ALIAS=country, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='US', $
```

```
GROUP=ADDRESSFINDEROPTIONS, ALIAS=addressFinderOptions, USAGE=A30,
 ACTUAL=A30, $
  FIELDNAME=DATASOURCE, ALIAS=dataSource, USAGE=A30, ACTUAL=A30,
   MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='GDT.Streets.US', $
  FIELDNAME=TOKEN, ALIAS=token, USAGE=A120, ACTUAL=A120,
   ACCESS_PROPERTY=(NEED_VALUE, AUTHTOKEN), $
  FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
   ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=FINDADDRESS,
   POSITION=__RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=findAddressResponse, USAGE=A1, ACTUAL=A1,
   ACCESS_PROPERTY=(INTERNAL), $
```

Example: **Passing Additional Parameters to an Authentication Operation**

You can pass an extra set of parameters to the authentication operation using XDEFAULT parameters to set the default values in the appropriate Master File.

Tip: To complete this task, you can use the Synonym Editor in the Web Console or the Data Management Console.

```
ENGINE SOAP SET CONNECTION_ATTRIBUTES belgR/:'https://secure.securex.be/
HRAWebservice/webservices/WSEmployee.asmx https://secure.securex.be/
HRAWebservice/webservices/WSEmployee.asmx?WSDL AUTH:belgium/
directauthenticate DBID="HRADemo01" LANGID="1"'
```

Master File fragment:

```
FILENAME=DIRECTAUTHENTICATE, SUFFIX=SOAP , $
SEGMENT=DIRECTAUTHENTICATE, SEGTYPE=S0, $


GROUP=DIRECTAUTHENTICATE, ALIAS=DirectAuthenticate, USAGE=A120,
   ACTUAL=A120, $
FIELDNAME=USERNAME, ALIAS=usrname, USAGE=A30, ACTUAL=A30,
   ACCESS_PROPERTY=(NEED_VALUE), $


FIELDNAME=PASSWORD, ALIAS=pwd, USAGE=A30, ACTUAL=A30,
   ACCESS_PROPERTY=(NEED_VALUE), $


FIELDNAME=DBID, ALIAS=dbId, USAGE=A30, ACTUAL=A30,
   ACCESS_PROPERTY=(NEED_VALUE), $
   XDEFAULT='Test', $


FIELDNAME=LANGID, ALIAS=langId, USAGE=A30, ACTUAL=A30,
   ACCESS_PROPERTY=(NEED_VALUE), $
   XDEFAULT='E', $


FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
   ACCESS_PROPERTY=(INTERNAL), $
```

```
SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=DIRECTAUTHENTICATE,
   POSITION=__RESPONSE, $

FIELDNAME=RESPONSE, ALIAS=DirectAuthenticateResponse, USAGE=A1,
   ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL), $

FIELDNAME=DIRECTAUTHENTICATERESULT, ALIAS=DirectAuthenticateResult,
  USAGE=A5, ACTUAL=A5, ACCESS_PROPERTY=(AUTHRESP), ACCEPT='true',
  REFERENCE=RESPONSE, PROPERTY=ELEMENT, $
```

Managing Web Services Metadata

When the server accesses a Web Services provider, it needs to know how to pass parameters to and accept responses from the Web service operation. For each operation the server will access, you create a synonym that describes the operation and the server mapping, which is provided through the WSDL file for the Web Service.

Creating Synonyms

A synonym defines a unique logical name (also known as an alias) for each Web Services operation. Synonyms are useful because:

- They insulate client applications from changes to the location and identity of a request. You can move or rename a request without modifying the client applications that use it. You need make only one change, redefining the request synonym on the server.
- They provide support for the extended metadata features of the server, such as virtual fields and security mechanisms.

Creating a synonym generates a Master File and an Access File. These are metadata that describe the Web Services request to the server.

Each synonym you create represents a single operation.

Note: Each synonym describes the Web Services operations of a particular provider. If the operation parameters are changed, the synonym must be recreated.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

Depending on the type of adapter you chose, one of the following options appears on the context menu.

- **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
- **Create metadata objects.** This opens the page for selecting synonym objects and properties.
- **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for Web Services

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

Filter Operations by Name

Leave this check box blank if you do not wish to filter the operations for which to create synonyms.

Select the check box to filter the operations. You are prompted an operation name. Enter a string for filtering the operations, inserting the wildcard character (%) as needed at the beginning and/or end of the string.

For example, enter: ABC% to select all operations whose names begin with the letters ABC; %ABC to select operations whose names end with the letters ABC; %ABC% to select operations whose names contain the letters ABC at the beginning, middle, or end.

Select Operations button

Click this button to select all or filtered operations.

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is unchecked, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Strict SOAP data type

By default, a SOAP request generated from a table request omits all elements tags for which screening values are not provided. If this is the behavior you want, leave the *Strict SOAP data type* box unchecked.

If you wish to pass empty element tags as well as those that contain values, click the *Strict SOAP data type* check box.

This option applies to both the body and header in a SOAP request.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

Overwrite Existing Synonyms

To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the *Overwrite existing synonyms* check box.

Note: The connected user must have operating system write privileges in order to recreate a synonym.

Default Synonym Name

This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.

To create synonyms

For all operations in the list, select the check box to the left of the Default Synonym Name column.

For specific operations, select the corresponding check boxes.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93.

Reference: Master File Attributes

Attribute	Description
PROPERTY	Indicates whether the field corresponds to an XML attribute or an XML element.

Attribute	Description
REFERENCE	Identifies the parent element of the field in the XML hierarchy, as defined by the Web Services XML schema document. The format is *segmentname.fieldname* where: *segmentname* Is the name of the Master File segment in which the field resides. *fieldname* Is the name of the field that corresponds to the parent element.

Reference: Access File Attributes

Attribute	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
CONNECTION	Indicates a previously declared connection. The syntax is: CONNECTION=*connection*
VERSION	Web Services version being used. The adapter supports Version 1.1.
ACTION	Web service operation being described.
OBJECT	Fully-qualified name of the Web Services request.
ENCODING	URL of the encoding schema.
TARGETNS	Target name space specified in the WSDL description of the operation. (This will either be global to the Web Service WSDL, or unique to the operation WSDL.)
STYLE	Value of the Web Services document WSDL style element (RPC or Document).

Attribute	Description
HEADER	Header group in the Master File, if one exists.
HEADERNS	Header namespace.
EMPTYTAGS	If ON, passes empty element tags, as well as those for which values have been provided. If OFF, the default, and element tags with values are passed. Applies to both the body and header in a SOAP request.

Using Static Joins

You can describe various views of the same physical XML document using Master Files and Access Files.

- In the Master File, you use REFERENCE attributes in field definitions to reflect physical relationships between tags in XML document hierarchies and PARENT attributes, which establish logical hierarchical relationships.
- In the Access File, you use the KEYFLD and IXFLD attributes to identify the XML tags that act as primary/foreign keys in the XML document hierarchies, which establish the logical join relationships. The parent field (foreign key) defined in KEYFLD supplies the value for cross-referencing; the descendant field (primary key) defined in IXFLD contains the corresponding value.

The adapter implements join matching values at run time.

You can use static Joins to create join relationships between hierarchically unrelated integral schema ComplexType definitions using any combination of data nodes.

Any XML tags belonging to these definitions can be used to create join pairs. In addition, you can multiply instances of the same physical segment to reflect logical join relationships as needed.

Using a Join can also provide a way to produce multiple executions of a Web service. However, the user cannot change the properties of the fields that describe the Web service input message. The original WSDL controls the acceptable data types for both the input and output. The Join must be to fields belonging to the root segment, and the ACCESS_PROPERTY parameter must be set to NEED_VALUE. The Join must also be unique to avoid looping.

Note: Static (embedded) Joins are not directly supported by the Create Synonym facility. To take advantage of this feature, after a synonym is generated you must modify its Master and Access Files. The Data Management Console Synonym Editor is the recommended tool for such modifications since it provides an easily identifiable segments hierarchy with drag and drop capability and a visual calculator for KEYFLD/IXFLD modifications. Using the DMC, you can quickly add all duplicate segments to the Master File, then delete any unneeded segments.

In some cases, schemas (such as those produced by Microsoft tools and reflecting relationships between the original MS SQL Server database tables) contain proper XML constraints (unique/key/keyref) for describing joins. You can use this information to modify the Master File:

- The elements *unique* and *key* define the primary key for the element (segment).
- The element *keyref* defines the *foreign* key for the element (segment).

For an example of how to use the Data Management Console to modify the Master File, see *Adding a Static Join From the Data Management Console Synonym Editor* on page 2622.

Example: Adding a Static Join From the Data Management Console Synonym Editor

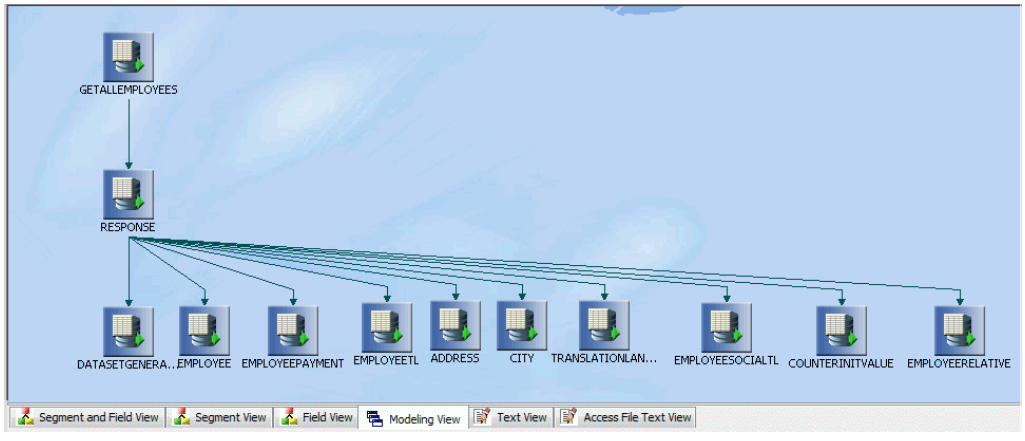
This example displays a fragment of the WSDL code from which a synonym called GetAllEmployees will be generated.

The schema shows a Constraint relationship between the Employee segment, keyed on the EID field, and the Employee Relative segment, keyed on ERID field.

```
<xs:unique msdata:PrimaryKey="true" name="Constraint1">
<xs:selector xpath=".//Employee" />
<xs:field xpath="EId" />
</xs:unique>

<xs:unique msdata:ConstraintName="Constraint1" msdata:PrimaryKey="true"
     name="EmployeeRelative_Constraint1">
  <xs:selector xpath=".//EmployeeRelative" />
  <xs:field xpath="ErId" />
  </xs:unique>
<xs:keyref msdata:ConstraintOnly="true"
     name="EmployeeRelative_Employee_EId_EId"
refer="mstns:Constraint1">
<xs:selector xpath=".//EmployeeRelative" />
<xs:field xpath="EId" />
</xs:keyref>
```

Working in the Data Management Console, you can quickly establish a Join relationship between the Employee and EmployeeRelative Segments in the synonym in order to be able to report against both segments at the same time.

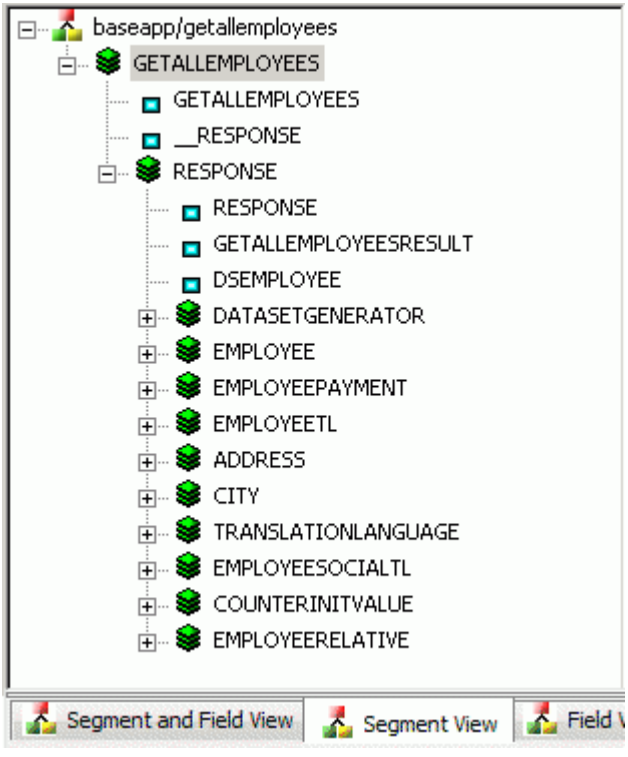
1. Open the DMC and open the Application Directory folder where the synonym is located.
2. Double-click the synonym *GetAllEmployees - Web Services* in the navigation pane, then click the *Modeling View* tab. The segment relationships are initially represented as follows:



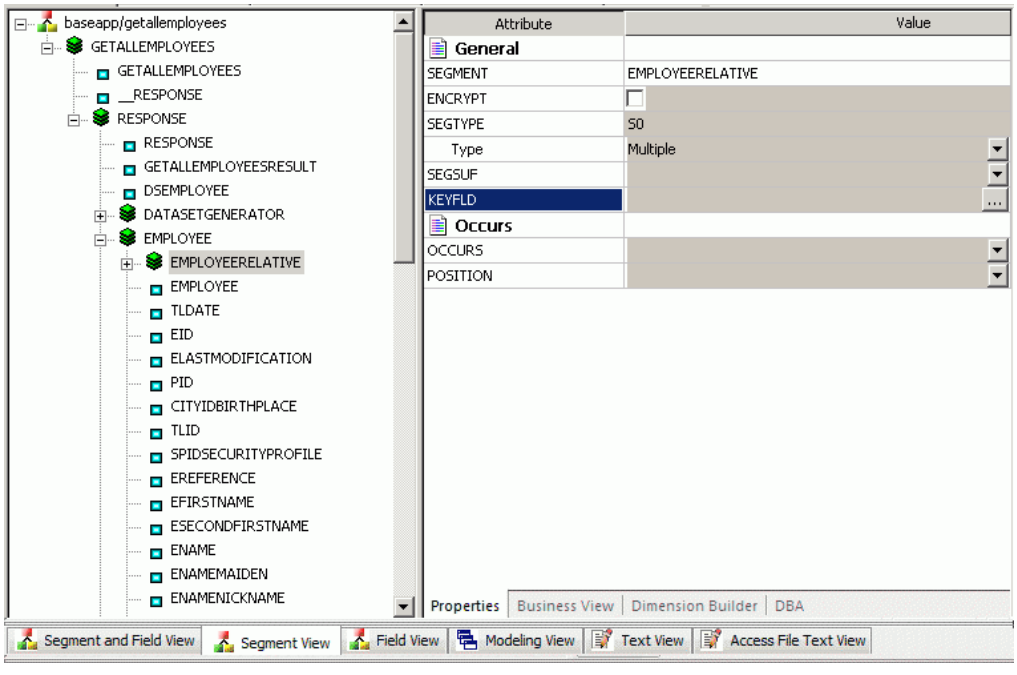
Notice that both Employee and EmployeeRelative are children of the Response segment in the generated synonym.

Your goal is to report against both segments at the same time, which is not something you can do using this structure. To accomplish that task, you must first make the EmployeeRelative segment a child of the Employee segment, an easy adjustment using the DMC Segment View.

3. Click the *Segment View* tab. You will see the following pane, in which Employee and Employee Relative are both children of segment Response, as shown in Modeling View:

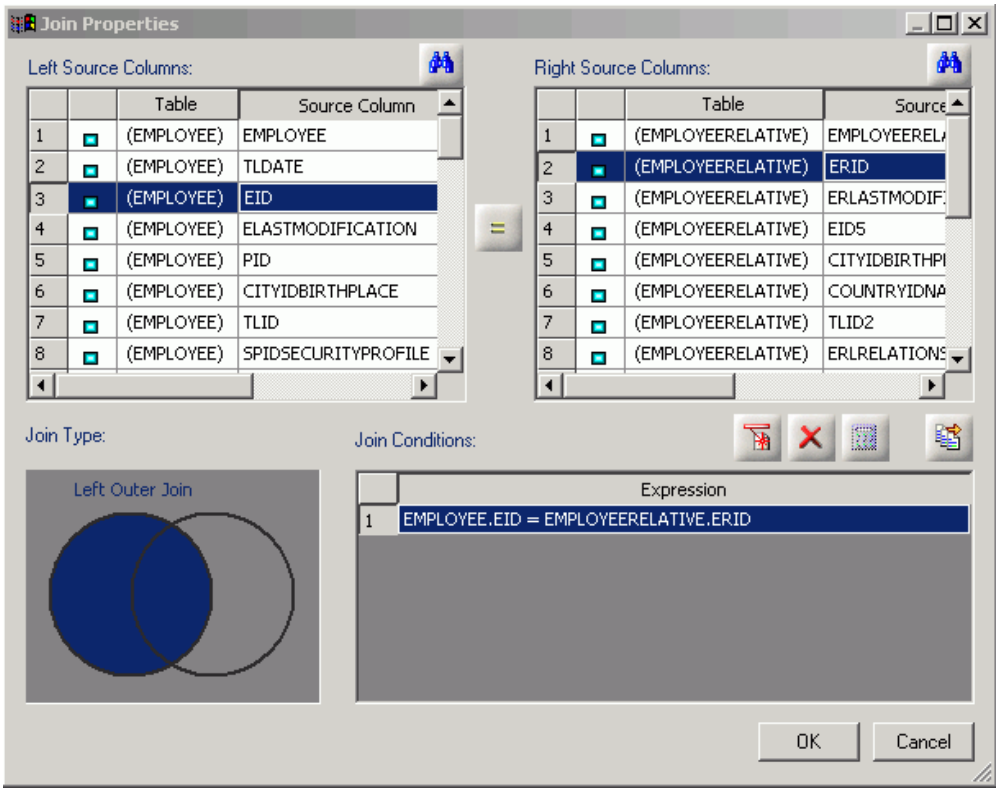


4. Drag the *EmployeeRelative* segment and drop it onto the Employee segment. EmployeeRelative is now a child of Employee, as shown in the revised Tree pane.



5. Next, you must set up the relationship that shows how the two segments are joined. Remember that in the schema the Employee segment and the EmployeeRelative segment are joined where EID = ERID.

   To mirror this relationship, click the *Ellipsis* (...) for *Keyfld* in the EmployeeRelative segment on the right pane. The Join Properties dialog box opens. This is where you will define the nature of the join relationship between the two segments.



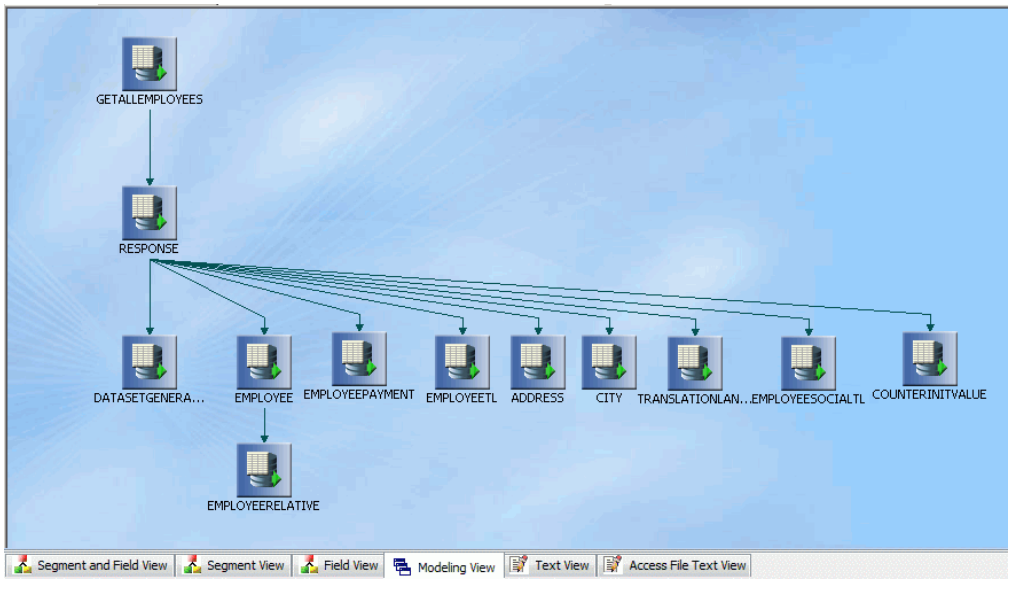
6. In the Venn Diagram box at the bottom left, click the outer portion of the left circle to specify a *Left Outer Join*. This option retrieves all records from the left-hand data source, along with any matching records from the right-hand data source. In this example, that means you will be able to display Employee values in the report even when the employees do not have any relatives.

   Tip: For definitions of other Join configurations that are available through the DMC, see *Join Options* on page 2627.

7. At the top of the dialog box, the Left Source Column displays fields from the Employee segment and the Right Source Column displays fields from the EmployeeRelative segment.

Select *EID* from the left side and *ERID* from the right side, then click the = sign between them.

8. Click *OK* to complete the Join and close the Join Properties dialog box.
9. Click the *Modeling View* tab to verify the modified structure in which EmployeeRelative is a child of Employee, providing the relationship you need to meet your reporting requirements.



Reference: Join Options

This chart defines the Join options that are available in the Venn Diagram box in the Data Management Console.

Inner Join indicates that two data sources are connected by an inner join and that only rows that appear in both tables used in the Join are extracted.

Left Outer Join indicates that two data sources are connected by a left outer join and that all rows are extracted from the left data source, as well as the columns from the right source that match.

Right Outer Join indicates that two data sources are connected by a right outer join and that all rows are extracted from the right data source, as well as the columns from the left source that match.

Full Outer Join indicates that two data sources are connected by a full outer join and that all rows are extracted from both data sources.

Cross Join indicates that two data sources are connected by a cross join, or Cartesian product of two tables. It consists of all possible pairs of rows between the two tables.

What Is a WSDL File?

A Web Services Description Language (WSDL) file is an XML document that describes a Web Service. A WSDL file contains the following key elements:

WSDL Element	Definition
definitions	Root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
types	Describes all the data types used between the client and server. WSDL is not tied exclusively to a specific typing system, but it uses the W3C XML Schema specification as its default choice. If the service uses only XML Schema built-in simple types, such as strings and integers, the types element is not required.
message	Describes a one-way message, whether it is a single message request or a single message response. It defines the name of the message and contains zero or more message part elements, which can refer to message parameters or message return values.
portType	Combines multiple message elements to form a complete one-way or round-trip operation. For example, a portType can combine one request and one response message into a single request/response operation, most commonly used in SOAP services. Note that a portType can (and frequently does) define multiple operations.
binding	Describes the concrete specifics of how the service will be implemented on the wire. WSDL includes built-in extensions for defining SOAP services, and SOAP-specific information therefore goes here.
service	Defines the address for invoking the specified service. Most commonly, this includes a URL for invoking the SOAP service.

Example: WSDL Segment for Generating a Master File for FindAddress Web Service

The following shows the elements for the input and output for the FindAddress Web Service function:

```
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com)-->
<wsdl:definitions
   xmlns:tns="http://arcweb.esri.com/v2"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:tme="http://www.themindelectric.com/"
xmlns:ns11="http://www.themindelectric.com/package/
com.esri.is.services.common.v2.geom/"
xmlns:ns12="http://www.themindelectric.com/package/com.esri.is.services.common.v2/"
xmlns:ns13="http://www.themindelectric.com/package/
com.esri.is.services.glue.v2.addressfinder/"
   xmlns:ns14="http://www.themindelectric.com/package/java.lang/"
   targetNamespace="http://arcweb.esri.com/v2" name="AddressFinder">
 <wsdl:types>

     <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.themindelectric.com/package/
com.esri.is.services.glue.v2.addressfinder/">
   <xsd:complexType name="Address">
    <xsd:sequence>
     <xsd:element name="houseNumber" nillable="true" type="xsd:string"/>
     <xsd:element name="street" nillable="true" type="xsd:string"/>
     <xsd:element name="intersection" nillable="true" type="xsd:string"/>
     <xsd:element name="city" nillable="true" type="xsd:string"/>
     <xsd:element name="state_prov" nillable="true" type="xsd:string"/>
     <xsd:element name="zone" nillable="true" type="xsd:string"/>
     <xsd:element name="country" nillable="true" type="xsd:string"/>
    </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="AddressFinderOptions">
    <xsd:sequence>
     <xsd:element name="dataSource" nillable="true" type="xsd:string"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:schema>
```

```
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.themindelectric.com/package/
com.esri.is.services.common.v2.geom/">
  <xsd:complexType name="Point">
   <xsd:sequence>
     <xsd:element name="x" type="xsd:double"/>
     <xsd:element name="y" type="xsd:double"/>
   <xsd:element name="coordinateSystem" nillable="true"
type="ns11:CoordinateSystem"/>
   </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CoordinateSystem">
   <xsd:sequence>
     <xsd:element name="projection" nillable="true" type="xsd:string"/>
     <xsd:element name="datumTransformation" nillable="true" type="xsd:string"/>
   </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Envelope">
   <xsd:sequence>
     <xsd:element name="minx" type="xsd:double"/>
     <xsd:element name="miny" type="xsd:double"/>
     <xsd:element name="maxx" type="xsd:double"/>
     <xsd:element name="maxy" type="xsd:double"/>
     <xsd:element name="coordinateSystem" nillable="true"
type="ns11:CoordinateSystem"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
```

```
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.themindelectric.com/package/
com.esri.is.services.common.v2/">
  <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
  <xsd:import namespace="http://www.themindelectric.com/package/
com.esri.is.services.common.v2.geom/"/>
  <xsd:complexType name="LocationInfo">
   <xsd:sequence>
    <xsd:element name="matchType" nillable="true" type="xsd:string"/>
    <xsd:element name="candidates" nillable="true" type="ns12:ArrayOfLocation"/>
    <xsd:element name="hasMore" type="xsd:boolean"/>
    <xsd:element name="errorCode" nillable="true" type="xsd:string"/>
   </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Location">
    <xsd:sequence>
     <xsd:element name="point" nillable="true" type="ns11:Point"/>
     <xsd:element name="description1" nillable="true" type="xsd:string"/>
     <xsd:element name="description2" nillable="true" type="xsd:string"/>
     <xsd:element name="score" type="xsd:double"/>
     <xsd:element name="matchType" nillable="true" type="xsd:string"/>
     <xsd:element name="type" nillable="true" type="xsd:string"/>
     <xsd:element name="locationExtent" nillable="true" type="ns11:Envelope"/>
    </xsd:sequence>
  </xsd:complexType>
   <xsd:complexType name="ArrayOfLocation">
    <xsd:complexContent>
     <xsd:restriction base="soapenc:Array">
       <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="ns12:Location[]"/>
     </xsd:restriction>
    </xsd:complexContent>
   </xsd:complexType>
  </xsd:schema>

  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.themindelectric.com/package/java.lang/">
     <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
     <xsd:complexType name="ArrayOfstring">
      <xsd:complexContent>
       <xsd:restriction base="soapenc:Array">
         <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"/>
       </xsd:restriction>
      </xsd:complexContent>
     </xsd:complexType>
  </xsd:schema>
 </wsdl:types>
```

```
    <message name="getAddress0In">
      <part name="point" type="ns11:Point"/>
      <part name="addressFinderOptions" type="ns13:AddressFinderOptions"/>
      <part name="token" type="xsd:string"/>
    </message>
    <message name="getAddress0Out">
      <part name="Result" type="ns13:Address"/>
    </message>

  <portType name="IAddressFinder">
     <operation name="getAddress" parameterOrder="point addressFinderOptions token">
       <documentation>Returns an address from an x,y-coordinate.</documentation>
        <input name="getAddress0In" message="tns:getAddress0In"/>
        <output name="getAddress0Out" message="tns:getAddress0Out"/>
     </operation>
  </portType>

  <binding name="IAddressFinder" type="tns:IAddressFinder">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getAddress">
    <soap:operation soapAction="getAddress" style="rpc"/>
     <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
     namespace="http://arcweb.esri.com/v2"/>
     </input>
     <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
  namespace="http://arcweb.esri.com/v2"/>
     </output>
  </operation>
  </binding>

  <service name="AddressFinder">
    <port name="IAddressFinder" binding="tns:IAddressFinder">
      <soap:address location="http://arcweb.esri.com/services/v2/AddressFinder"/>
    </port>
  </service>
</wsdl:definitions>
```

The following Master File is generated from the Create Synonym process:

```
FILENAME=M6ILO, SUFFIX=SOAP    , $
  SEGMENT=GETADDRESS, SEGTYPE=S0, $
   GROUP=POINT, ALIAS=point, USAGE=A76, ACTUAL=A100, $
    FIELDNAME=X, ALIAS=x, USAGE=D20.2, ACTUAL=A20, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=Y, ALIAS=y, USAGE=D20.2, ACTUAL=A20, ACCESS_PROPERTY=(NEED_VALUE), $
   GROUP=COORDINATESYSTEM, ALIAS=coordinateSystem, USAGE=A60, ACTUAL=A60, $
    FIELDNAME=PROJECTION, ALIAS=projection, USAGE=A30, ACTUAL=A30,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=DATUMTRANSFORMATION, ALIAS=datumTransformation, USAGE=A30, ACTUAL=A30,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
   GROUP=ADDRESSFINDEROPTIONS, ALIAS=addressFinderOptions, USAGE=A30, ACTUAL=A30, $
    FIELDNAME=DATASOURCE, ALIAS=dataSource, USAGE=A30, ACTUAL=A30,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=TOKEN, ALIAS=token, USAGE=A30, ACTUAL=A30,
      ACCESS_PROPERTY=(NEED_VALUE), $
    FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX, ACCESS_PROPERTY=(INTERNAL), $

  SEGMENT=RESPONSE, SEGTYPE=S0,SEGSUF=XML     , PARENT=GETADDRESS,
POSITION=__RESPONSE, $
    FIELDNAME=RESPONSE, ALIAS=getAddress0Out, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL), $
  SEGMENT=RESULT, SEGTYPE=S0, PARENT=RESPONSE, $
    FIELDNAME=RESULT, ALIAS=Result, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=RESPONSE, PROPERTY=ELEMENT,  $
    FIELDNAME=HOUSENUMBER, ALIAS=houseNumber, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=STREET, ALIAS=street, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=INTERSECTION, ALIAS=intersection, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=CITY, ALIAS=city, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=STATE_PROV, ALIAS=state_prov, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=ZONE, ALIAS=zone, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=COUNTRY, ALIAS=country, USAGE=A30, ACTUAL=A30,
      MISSING=ON,
      REFERENCE=RESULT, PROPERTY=ELEMENT,  $
```

The following Access File is generated from the Create Synonym process:

```
SEGNAME=GETADDRESS, CONNECTION=CON06, VERSION=1.1, OBJECT=getAddress,
  ACTION=getAddress, ENCODING=http://schemas.xmlsoap.org/soap/encoding/,
  TARGETNS=http://arcweb.esri.com/v2, STYLE=RPC, $
 ID=ns11, NS=http://www.themindelectric.com/package/
    com.esri.is.services.common.v2.geom/   , $
  FIELD=point, TYPE=Point, NS_ID=ns11, $
  FIELD=coordinateSystem, TYPE=CoordinateSystem, NS_ID=ns11, $
 ID=ns13, NS=http://www.themindelectric.com/package/
    com.esri.is.services.glue.v2.addressfinder/   , $
  FIELD=addressFinderOptions, TYPE=AddressFinderOptions, NS_ID=ns13, $
```

You can use the following procedure to retrieve data from the FindAddress Master File:

```
SET ALL=ON
TABLE FILE FINDADDRESS
PRINT LOCATION.X LOCATION.Y
WHERE FINDADDRESS.HOUSENUMBER EQ '27'
WHERE FINDADDRESS.STREET EQ 'Stevenson Drive'
WHERE FINDADDRESS.ZONE EQ '07746'
WHERE FINDADDRESS.DATASOURCE EQ 'GDT.Streets.US'
END
```

Example: WSDL Document With SOAP Header

The Create Synonym process generates an extra group in the root segment to describe the SOAP HEADER layout, as defined in the WSDL document.

The following WSDL describes a SOAP header:

Binding section:

```
<input>
  <soap:body use="literal"/>
  <soap:header part="header" message="tns:jobsHeader" use="literal"/>
</input>
```

Message description:

```
<message name="jobsHeader">
  <part element="tns:ibsinfo" name="header"/>
</message>
```

Types section:

```
<xs:element name="ibsinfo">
  <xs:complexType>
      <xs:sequence>
          <xs:element type="xs:string" name="service"/>
          <xs:element type="xs:string" name="method"/>
          <xs:element type="xs:string" name="license"/>
          <xs:element type="xs:string" minOccurs="0" name="disposition"/>
          <xs:element type="xs:string" minOccurs="0" name="Username"/>
          <xs:element type="xs:string" minOccurs="0" name="Password"/>
          <xs:element type="xs:string" minOccurs="0" name="language"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>
```

The resulting Master File contains:

```
SEGMENT=LOCATION, SEGTYPE=S0, $
  GROUP=HEADER, ALIAS=Header, USAGE=A210, ACTUAL=A210, $
  GROUP=IBSINFO, ALIAS=ibsinfo, USAGE=A210, ACTUAL=A210, $
   FIELDNAME=SERVICE, ALIAS=service, USAGE=A30, ACTUAL=A30,
      ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=METHOD, ALIAS=method, USAGE=A30, ACTUAL=A30,
      ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=LICENSE, ALIAS=license, USAGE=A30, ACTUAL=A30,
      ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=DISPOSITION, ALIAS=disposition, USAGE=A30, ACTUAL=A30,
     ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=USERNAME, ALIAS=Username, USAGE=A30, ACTUAL=A30,
     ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=PASSWORD, ALIAS=Password, USAGE=A30, ACTUAL=A30,
     ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=LANGUAGE, ALIAS=language, USAGE=A30, ACTUAL=A30,
     ACCESS_PROPERTY=(NEED_VALUE), $
```

The resulting Access File contains:

```
SEGNAME=LOCATION, CONNECTION=PSIBS, VERSION=1.1, OBJECT=LOCATION,
  ACTION=EFREM.LOCATIONRequest@test@@, HEADER=HEADER,
  TARGETNS=urn:iwaysoftware:ibse:jul2003:LOCATION,
  HEADERNS=urn:schemas-iwaysoftware-com:iwse, STYLE=DOCUMENT, $
```

Note that the SOAP header is generated in the SOAP request if a value is provided (either explicitly in the request or implicitly via default values in the Master File) for *at least one* of the header components. If no header values are available, the SOAP header is not generated.

Example: WSDL Document With Output Headers

The Create Synonym process generates extra output segment(s) to describe the header data returned by a SOAP request. Segments describing the header and body parts of the SOAP response become children of the response segment and siblings to each other.

WSDL fragment

```
<operation name="TrackMessagesBulk">
< soap:operation soapAction="http://www.strikeiron.com/TrackMessagesBulk"
       style="document" />
  <input>
    <soap:body use="literal" />
  </input>
 <output>
   <soap:body use="literal" />
    <soap:header message="s0:TrackMessagesBulkResponseInfo"
          part="ResponseInfo" use="literal" />
  </output>
</operation>
```

Generated Master File

```
SEGMENT=RESPONSE, SEGTYPE=S0, SEGSUF=XML , PARENT=TRACKMESSAGESBULK
      POSITION=__RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=Envelope, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL), $
  FIELDNAME=HEADER, ALIAS=Header, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL), $
  FIELDNAME=BODY, ALIAS=Body, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL), $
SEGMENT=HEADER, SEGTYPE=S0, PARENT=RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=ResponseInfo, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL), REFERENCE=HEADER, $
  FIELDNAME=RESPONSECODE, ALIAS=ResponseCode, USAGE=I11, ACTUAL=A11,
      REFERENCE=HEADER.RESPONSE, PROPERTY=ELEMENT,  $
  FIELDNAME=RESPONSE1, ALIAS=Response, USAGE=A30, ACTUAL=A30,
      REFERENCE=HEADER.RESPONSE, PROPERTY=ELEMENT,  $
SEGMENT=BODY, SEGTYPE=S0, PARENT=RESPONSE, $
  FIELDNAME=RESPONSE, ALIAS=TrackMessagesBulkResponse, USAGE=A1,
      ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL), REFERENCE=BODY, $
  FIELDNAME=TRACKMESSAGESBULKRESULT, ALIAS=TrackMessagesBulkResult,
      USAGE=A1, ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=BODY.RESPONSE, PROPERTY=ELEMENT,  $
```

Reference: Multiple Headers

The Adapter for Web Services converts multiple header body definitions in the WSDL document into Master File components. Each input header body becomes a separate GROUP description for the input SOAP segment. For each output header body, a special field description is created in the response segment.

Example: WSDL Document With Mixed Content

Complex data types with the MIXED property are supported in the SOAP response document.

```
<xsd:element name="letterBody">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="salutation">
                <xsd:complexType mixed="true">
                    <xsd:sequence>
                        <xsd:element name="name" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="quantity" type="xsd:positiveInteger"/>
            <xsd:element name="productName" type="xsd:string"/>
            <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

The resulting Master File contains:

```
SEGMENT=LETTERBODY, SEGTYPE=S0, $
 FIELDNAME=LETTERBODY, ALIAS=letterBody, USAGE=A50, ACTUAL=A50, $
 FIELDNAME=SALUTATION, ALIAS=salutation, USAGE=A50, ACTUAL=A50,
  REFERENCE=LETTERBODY, PROPERTY=ELEMENT, $
 FIELDNAME=NAME, ALIAS=name, USAGE=A50, ACTUAL=A50,
  REFERENCE=SALUTATION, PROPERTY=ELEMENT, $
 FIELDNAME=QUANTITY, ALIAS=quantity, USAGE=P32, ACTUAL=A32,
  REFERENCE=LETTERBODY, PROPERTY=ELEMENT, $
 FIELDNAME=PRODUCTNAME, ALIAS=productName, USAGE=A50, ACTUAL=A50,
  REFERENCE=LETTERBODY, PROPERTY=ELEMENT, $
 FIELDNAME=SHIPDATE, ALIAS=shipDate, USAGE=YYMD, ACTUAL=A10, MISSING=ON,
  REFERENCE=LETTERBODY, PROPERTY=ELEMENT, $
```

The Adapter for Web Services does not preserve the original sequence of text components and descendant elements. The following SOAP response document and report illustrate this limitation:

```
<letterBody>
    <salutation>Dear Mr.
        <name>Robert Smith</name>.
    </salutation>
    Your order of
    <quantity>1</quantity>
    <productName>BabyMonitor</productName>
    shipped from our warehouse on
    <shipDate>1999-05-21</shipDate>.
</letterBody>
```

```
LETTERBODY       SALUTATION NAME           QUANTITY PRODUCTNAME SHIPDATE
----------       ---------- ----           -------- ----------- --------

Your order of shipped from our warehouse on .  Dear Mr. . Robert Smith
1        BabyMonitor 1999/05/21
```

Example: WSDL Document With Abstract Input Data Type

The adapter generates an input field with ACTUAL and USAGE formats of A1. The ACCESS property for this field is initially set to INTERNAL to prevent a front-end tool from requesting an input value. A user can modify the definition manually, if necessary.

Message description:

```
<message name="ivpIn">
    <part element="m1:ivp" name="parameters"/>
</message>
Input message definition:
<xs:element name="ivp">
    <xs:complexType>
        <xs:sequence/>
    </xs:complexType>
</xs:element>
```

The resulting Master File contains:

```
FIELDNAME=IVP, ALIAS=ivp, USAGE=A1, ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL), $
```

Mapping Attributes and Default Values

Element attributes are supported as part of the service input message definition:

- Restriction values are stored in the corresponding field definition of the Master File as an ACCEPT list.
- The default value is stored in the XDEFAULT parameter.

Example: WSDL Document With Mapped Attributes and Default Values

```
<xs:element name="component">
  <xs:complexType>
     <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute use="optional" default="browse" name="perform">
             <xs:simpleType>
               <xs:restriction base="xs:string">
                   <xs:enumeration value="browse"/>
                   <xs:enumeration value="insert"/>
                   <xs:enumeration value="update"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
        </xs:extension>
     </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

The resulting Master Files contains:

```
FIELDNAME=COMPONENT, ALIAS=component, USAGE=A30, ACTUAL=A30,
 ACCESS_PROPERTY=(NEED_VALUE), $

FIELDNAME=PERFORM, ALIAS=perform, USAGE=A6, ACTUAL=A6,
 ACCESS_PROPERTY=(NEED_VALUE),

XDEFAULT='browse',
ACCEPT='browse' OR 'insert' OR 'update',
REFERENCE=COMPONENT, PROPERTY=ATTRIBUTE, $
```

Note that if no explicit value is provided for the field, the XDEFAULT value is used when the SOAP request is generated.

Using Arrays as Input Values

The Adapter for Web Services supports the use of arrays as input values. An array is described in the Master File using the GROUP definition with ACCESS_PROPERTY=ARRAY_ITEM. GROUP describes one instance of an object. You can provide input values for the array of objects using the following methods:

- Specify lazy OR screening conditions for individual fields in a GROUP.
- Specify lazy OR screening conditions for the GROUP itself. (Values for the individual GROUP fields must be separated by /).
- Read values from a file specified as follows:

  ```
  (IF group EQ (ddname))
  ```

  Only the GROUP describing an array can be used in an in-file construct. For example:

  ```
  FILEDEF CARRAY DISK C:\Users\.........\countryarray.ftm
  .
  .
  .
  IF COUNTRY  EQ (CARRAY)
  ```

Note that only the last screening value is reported for an input array component.

Example: Using Arrays as Input Values

In the following example a segment defining the structure of an ESRI SOAP request retrieves a list of coordinates for the set of addresses. GROUP ADDRESSES are used to define one object item from the ADDRESSES array. The GROUP definition is extended by adding ACCESS_PROPERTY=ARRAY_ITEM and providing input values for the fields that comprise the array.

Master File

```
SEGMENT=FINDADDRESS, SEGTYPE=S0, $
 GROUP=ADDRESSES, ALIAS=addresses, USAGE=A210, ACTUAL=A210,
 ACCESS_PROPERTY=ARRAY_ITEM, $
   FIELDNAME=HOUSENUMBER, ALIAS=houseNumber, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='2815', $
   FIELDNAME=STREET, ALIAS=street, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='PRAIRIE AVE.', $
   FIELDNAME=INTERSECTION, ALIAS=intersection, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
   FIELDNAME=CITY, ALIAS=city, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='MIAMI BEACH', $
   FIELDNAME=STATE_PROV, ALIAS=state_prov, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='FL', $
   FIELDNAME=ZONE, ALIAS=zone, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='33140', $
   FIELDNAME=COUNTRY, ALIAS=country, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), XDEFAULT='US', $
 GROUP=ADDRESSFINDEROPTIONS, ALIAS=addressFinderOptions, USAGE=A30,
 ACTUAL=A30, $
   FIELDNAME=DATASOURCE, ALIAS=dataSource, USAGE=A30, ACTUAL=A30,
     MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE),
     XDEFAULT='GDT.Streets.US', $
   FIELDNAME=TOKEN, ALIAS=token, USAGE=A120, ACTUAL=A120,
     ACCESS_PROPERTY=(NEED_VALUE, AUTHTOKEN), $
   FIELDNAME=__RESPONSE, USAGE=TX80L, ACTUAL=TX,
     ACCESS_PROPERTY=(INTERNAL), $
```

Note: Although not illustrated in this example, you can use the ACCESS_PROPERTY=ARRAY_ITEM value to create nested GROUP definitions in which an inner GROUP object serves as an array inside an array. Using nested arrays, you can describe virtually any hierarchical object.

Screening on Leaf Components

Following are illustrations of the two variations on lazy OR screening on leaf components.

Note that it is not necessary to provide values for all fields in a GROUP since XML array item objects can be built with the subset of object components. However, the two screening methods have different conventions for dealing with missing values, as illustrated in the following examples.

Both of these requests produce identical elements in the MATCH array.

Explicit screening by field

When you are specifying lazy OR screening conditions for individual fields in a GROUP, the number of values and their positions must be the same for all field in the array GROUP.

Notice that each field below has three values. When an array is generated from this input, the first values in each line will be aligned, the seconds values will be aligned, and the third values will be aligned. Therefore, if values are omitted or misplaced, the alignment will be incorrect.

```
IF HOUSENUMBER EQ '2815' OR 'Two' OR '1250'
IF STREET EQ 'PRAIRIE AVE.' OR 'Penn Plaza' OR 'Broadway'
IF CITY EQ 'MIAMI BEACH' OR 'NY' OR 'NY'
IF STATE_PROV EQ 'FL' OR 'NY' OR 'NY'
IF ZONE EQ '33140' OR '10121' OR '10001'
IF COUNTRY EQ 'US' OR 'US' OR 'US'
```

Note that there are no values for the INTERSECTION field in this screening request. When specifying individual fields explicitly you can simply omit any fields you are not interested in including in the array.

Screening for a whole group

When you specify lazy OR screening conditions for the GROUP, values for individual fields in the GROUP must be separated by a slash (/). Missing values like INTERSECTION must be accounted for using two consecutive slashed (//) on each line of the screening request.

```
IF COUNTRY EQ '2815/PRAIRIE AVE.//MIAMI BEACH/FL/33140/US' OR
              'Two/Penn Plaza//NY/NY/10121' OR
              '1250/Broadway//NY/NY/10001'
```

In this example, INTERSECTION is represented by // after STREET and before CITY.

Using Arrays as Input Values for XML Elements and Their Attributes

You can apply standard Web Services array methodology to XML attribute values that are associated with parent XML array element values. The Parent array element is described with the ARRAY_ITEM value in the ACCESS_PROPERTY parameter. Associated attributes are linked to the parent element using the REFERENCE parameter, with the PROPERTY parameter set to ATTRIBUTE. You can supply input values for the element and attributes as you would for a regular array.

Example: Using Arrays as Input Values for an XML Element and its Attributes

Master File fragment:

```
FIELDNAME=KEY, ALIAS=key, USAGE=A30, ACTUAL=A30,
  ACCESS_PROPERTY=(ARRAY_ITEM), $
FIELDNAME=NAME, ALIAS=name, USAGE=A30, ACTUAL=A30,
  ACCESS_PROPERTY=(NEED_VALUE), REFERENCE=KEY, PROPERTY=ATTRIBUTE,  $
```

Request fragment with screening statements:

```
IF KEY EQ 'SHARE' OR 'BAE01A'
IF NAME EQ 'Setid' OR 'Location'
```

Generated SOAP request:

The screening statements yield an array of key/name occurrences in the soap request:

```
<SOAP-ENV:Body>
    <m:GetQuote xmlns:m="http://ws.cdyne.com/">
       <m:key xsi:type="xsd:string" name="Setid"  >SHARE</m:key>
       <m:key xsi:type="xsd:string" name="Location" >BAE01A</m:key>
   </m:GetQuote>
</SOAP-ENV:Body>
```

Modifying Namespace Qualifiers in WSDL Schemas

Namespace qualifiers are generated in the SOAP request document in accordance with the WSDL schemas. You can modify qualification rules in the Access File to affect how the SOAP XML request is built.

Qualification control information is stored in the Access File on the segment, namespace (Id), and element/attribute (Master File field) levels.

- Segment level qualification governs qualifiers for all element/attributes in the Body/Header, respectively, belonging to the target namespace. Segment level qualification information is taken from the schema level for the target namespace.
- ID level qualification governs qualifiers for all element/attributes in the described namespace. ID level qualification information is taken from the particular namespace corresponding to the referenced schema.
- Field level qualification governs qualifiers for the particular element/attribute and overwrites Segment/ID settings. Field level qualification is taken from the particular schema element type definition.

New parameters are introduced in the Access File in both the Segment and ID records:

- ELEMFORM and ATTRFORM (for the SOAP Body).
- HELEMFORM and HATTRFORM (for the SOAP Header).

A new FORM parameter is introduced in the Field record that describes the element or attribute.

Acceptable values for the new parameters are *qualified* and *unqualified*.

The default behavior (full qualification for all elements in the Header and Body, and no qualification for the attributes) is applied when no new parameters are found in the Access File.

See the *XML Schema Part 0: Primer*, which you can access at *http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/#NS*, for details about *Advanced Concepts: Namespaces, Schemas & Qualification*.

Data Type Support

The following table lists how the server maps XSD data types in a Master File.

XSD Data Type	USAGE	ACTUAL
string	A30	A30
double	D20.2	A20
float	F15.2	A15
decimal	P20.3	A20
int	I11	A11
short	I6	A6
long	P20	A20
boolean	A5	A5
dateTime	HYYMDm For related information, see *Date-Time Processing* on page 2644.	A35
time	HHISsm	A15
date	YYMD	A10
gYearMonth	HYYM	A8
gYear	HYY	A5
gMonthDay	HMD	A6
gDay	HD	A3
gMonth	HM	A4
integer	P33	A33
nonPositiveInteger	P33	A33
negativeInteger	P33	A33

XSD Data Type	USAGE	ACTUAL
nonNegativeInteger	P32	A32
unsignedLong	P20	A20
unsignedInt	P10	A10
unsignedShort	I5	A5
positiveInteger	P32	A32
byte	I4	A4
unsignedByte	I4	A4
normalizedString	A30	A30
token	A30	A30
Name	A30	A30
NMTOKEN	A30	A30
ID	A30	A30
hexBinary	A30	A30
language	A30	A30
anyURI	A30	A30
QName	A30	A30

Date-Time Processing

Date-time formats can produce output values and accept input values that are compatible with the ISO 8601:2000 date-time notation standard.

Syntax: How to Enable ISO Standard Date-Time Notation

```
SET DTSTANDARD = {OFF|ON|STANDARD|STANDARDU}
```

where:

OFF

Does not provide compatibility with the ISO 8601:2000 date-time notation standard.

ON|STANDARD

Enables recognition and output of the ISO standard formats, including use of T as the delimiter between date and time, use of period or comma as the delimiter of fractional seconds, use of Z at the end of "universal" times, and acceptance of inputs with time zone information. STANDARD is a synonym for ON.

STANDARDU

Enables ISO standard formats (like STANDARD) and also, where possible, converts input strings to the equivalent "universal" time (formerly known as "Greenwich Mean Time"), thus enabling applications to store all date-time values in a consistent way.

Capturing a SOAP Request Using FILEDEF SOAPTSCQ in a Procedure

You can use the command FILEDEF SOAPTSCQ within a procedure to capture the SOAP request being generated and store it in a designated location in the file system. Since the SOAP request itself is not actually executed, 0 records are returned in the report.

Example: Using FILEDEF SOAPTSCG to Capture a SOAP Request

In this example, the procedure reports against a Web Service operation that has the two parameters: SYMBOL and USERNAME. The generated SOAP request is stored in a file called SOAPrequest.xml.

```
FILEDEF SOAPTSCQ DISK C:\IBI\APPS\XIGNITE\SOAPrequest.xml
TABLE FILE XQUOTES_GETQUOTE
PRINT SYMBOL1 LAST
WHERE SYMBOL EQ `IBM`
WHERE USERNAME EQ `myusername`
END
```

The following is the content of SOAPrequest.xml after the procedure is executed:

```
<soap_tscq>
<soap_url>http://www.xignite.com/xQuotes.asmx</soap_url>
<soap_action>http://www.xignite.com/services/GetQuote</soap_action>
<soap_body>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/
2001/XMLSchema"
  xmlns:tns="http://www.xignite.com/services/">
<SOAP-ENV:Header>
<m:Header xmlns:m="http://www.xignite.com/services/">
<m:Username xsi:type="xsd:string">myusername</m:Username>
  </m:Header>
  </SOAP-ENV:Header>
<SOAP-ENV:Body>
<m:GetQuote xmlns:m="http://www.xignite.com/services/">
<m:Symbol xsi:type="xsd:string">IBM</m:Symbol>
  </m:GetQuote>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
  </soap_body>
  </soap_tscq>
```

Chapter 108

Using the Adapter for WebFOCUS Client REST

The Adapter for WebFOCUS Client REST integrates with the WebFOCUS Client through WebFOCUS RESTful Web Services. Explicit and Password Passthru authentication to WebFOCUS is handled by configuring the connection to the adapter. A successful authentication returns a Cross-Site Request Forgery (CSRF) token that is automatically passed on each WebFOCUS RESTful Web Service call.

Once you add a connection for the Adapter for WebFOCUS Client REST, you can use the connection to create an application in the Server Web Console that is linked to an existing WebFOCUS Repository. For information, see the *Server Administration* manual.

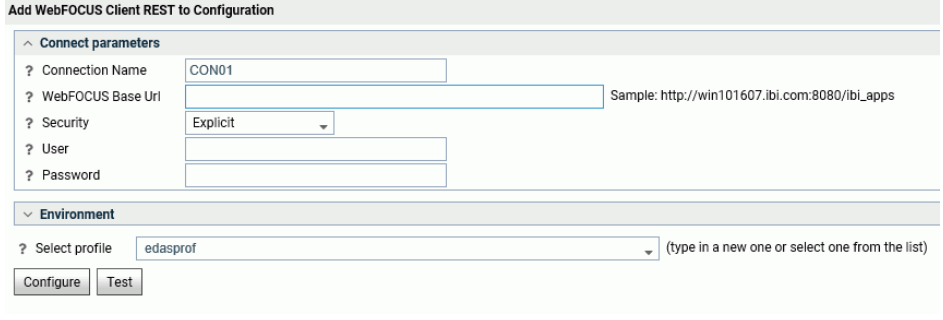
When you right-click the connection and click *Show DBMS objects*, you can then click *Create Synonym(s) and Examples*, which loads a set of examples that include Master Files, Access Files, and Procedures. Each of these examples performs a specific WebFOCUS function, such as security maintenance, ReportCaster scheduling, and folder maintenance.

In this chapter:

Configuring the Adapter for WebFOCUS Client REST

1. Click *Connect to Data* on the Web Console sidebar.
2. Right-click the WebFOCUS Client REST adapter on the Available list and click *Configure*.

   The Add Connection page opens, as shown in the following image.



3. Configure the following parameters.

   `Connection Name`
   Is a name for the connection.

   `WebFOCUS Base Url`
   Is the URL used to access WebFOCUS, in the form http://*computername*:*port*/ibi_apps.

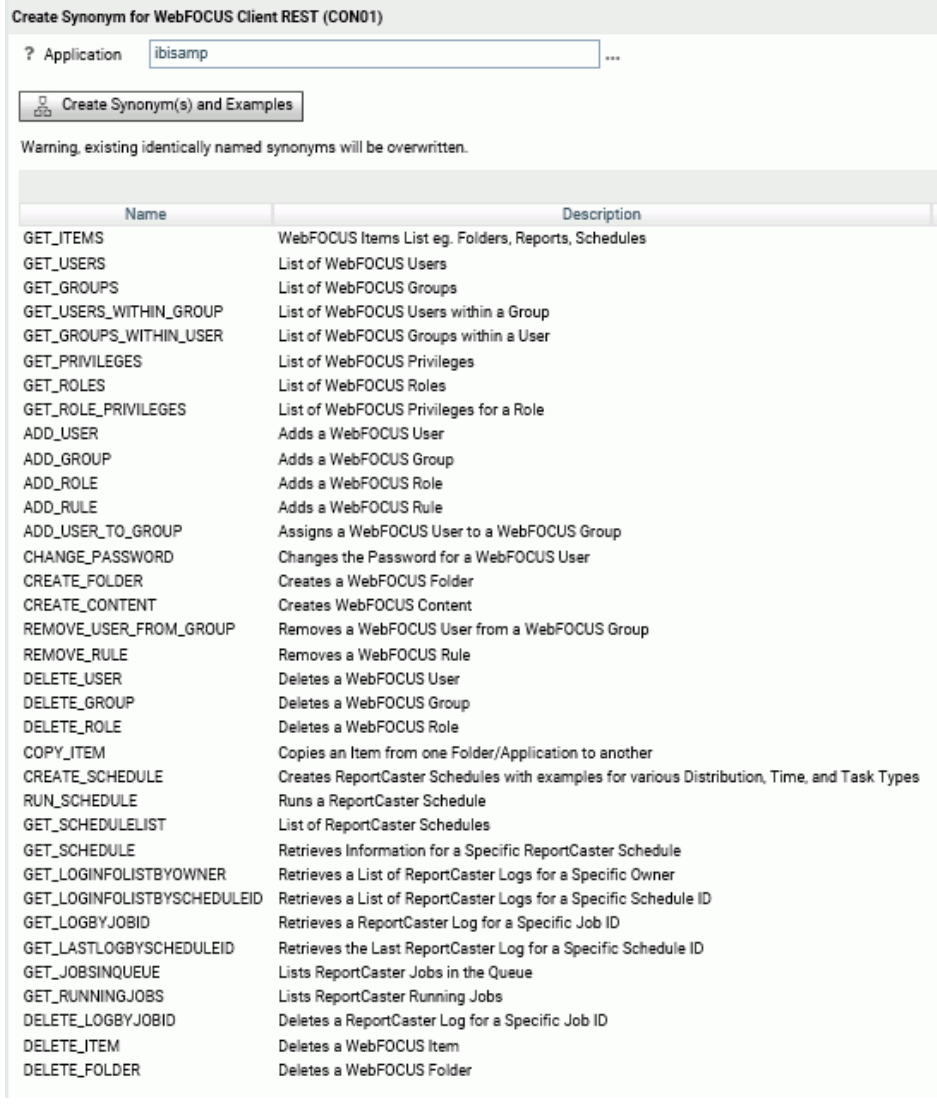
   `Security`
   Select Explicit or Password Passthru. For Explicit authentication, enter the WebFOCUS user ID and password.

4. Click *Test* to verify the connection parameters.
5. When you have verified the connection parameters, click *Configure*.

Creating Synonyms and Examples for the Adapter for WebFOCUS Client REST

1. Right-click a connection for the Adapter for WebFOCUS Client REST and click *Show DBMS objects*.

The Create Synonym for WebFOCUS Client REST page opens, as shown in the following image.



Name	Description
GET_ITEMS	WebFOCUS Items List eg. Folders, Reports, Schedules
GET_USERS	List of WebFOCUS Users
GET_GROUPS	List of WebFOCUS Groups
GET_USERS_WITHIN_GROUP	List of WebFOCUS Users within a Group
GET_GROUPS_WITHIN_USER	List of WebFOCUS Groups within a User
GET_PRIVILEGES	List of WebFOCUS Privileges
GET_ROLES	List of WebFOCUS Roles
GET_ROLE_PRIVILEGES	List of WebFOCUS Privileges for a Role
ADD_USER	Adds a WebFOCUS User
ADD_GROUP	Adds a WebFOCUS Group
ADD_ROLE	Adds a WebFOCUS Role
ADD_RULE	Adds a WebFOCUS Rule
ADD_USER_TO_GROUP	Assigns a WebFOCUS User to a WebFOCUS Group
CHANGE_PASSWORD	Changes the Password for a WebFOCUS User
CREATE_FOLDER	Creates a WebFOCUS Folder
CREATE_CONTENT	Creates WebFOCUS Content
REMOVE_USER_FROM_GROUP	Removes a WebFOCUS User from a WebFOCUS Group
REMOVE_RULE	Removes a WebFOCUS Rule
DELETE_USER	Deletes a WebFOCUS User
DELETE_GROUP	Deletes a WebFOCUS Group
DELETE_ROLE	Deletes a WebFOCUS Role
COPY_ITEM	Copies an Item from one Folder/Application to another
CREATE_SCHEDULE	Creates ReportCaster Schedules with examples for various Distribution, Time, and Task Types
RUN_SCHEDULE	Runs a ReportCaster Schedule
GET_SCHEDULELIST	List of ReportCaster Schedules
GET_SCHEDULE	Retrieves Information for a Specific ReportCaster Schedule
GET_LOGINFOLISTBYOWNER	Retrieves a List of ReportCaster Logs for a Specific Owner
GET_LOGINFOLISTBYSCHEDULEID	Retrieves a List of ReportCaster Logs for a Specific Schedule ID
GET_LOGBYJOBID	Retrieves a ReportCaster Log for a Specific Job ID
GET_LASTLOGBYSCHEDULEID	Retrieves the Last ReportCaster Log for a Specific Schedule ID
GET_JOBSINQUEUE	Lists ReportCaster Jobs in the Queue
GET_RUNNINGJOBS	Lists ReportCaster Running Jobs
DELETE_LOGBYJOBID	Deletes a ReportCaster Log for a Specific Job ID
DELETE_ITEM	Deletes a WebFOCUS Item
DELETE_FOLDER	Deletes a WebFOCUS Folder

2. Enter an application name for the synonyms and examples in the Application text box, or click the ellipsis (...) to browse to an application.
3. Click *Create Synonym(s) and Example*s.

   A set of synonyms and a folder with FOCEXECs is created in the application you selected.

For example, running the get_groups FOCEXEC with the default WebFOCUS groups configured produces the output shown in the following image.

WebFOCUS
List of Groups
For Parent Group Path: IBFS:/SSYS/GROUPS

Group Name	Group Description	Group Path
EVERYONE	All defined users	IBFS:/SSYS/GROUPS/EVERYONE
Administrators	Administrators	IBFS:/SSYS/GROUPS/Administrators
Anonymous	Anonymous users	IBFS:/SSYS/GROUPS/Anonymous
SelfServiceDevelopers	Developers of content for EDA and WEB only	IBFS:/SSYS/GROUPS/SelfServiceDevelopers
Managers	Managers	IBFS:/SSYS/GROUPS/Managers

Chapter 109

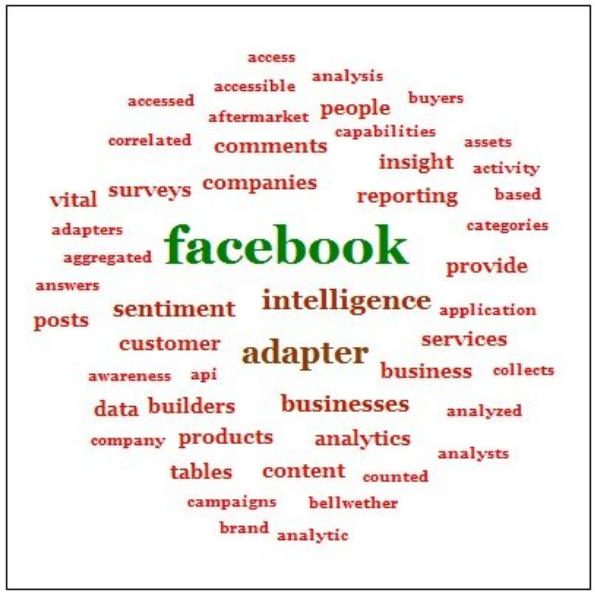
Using the Adapter for Words Analysis

This section describes how to configure the Words Analysis Adapter.

In this chapter:

Words Analysis Adapter Overview

The Words Analysis Adapter counts the occurrences of each word within textual data. It includes a Stopwords file, which can be modified, to define the words to exclude from the analysis. The results can be displayed in a Tabular report or graph. A tag cloud graph is a popular choice for analyzing the occurrences of words within textual data.



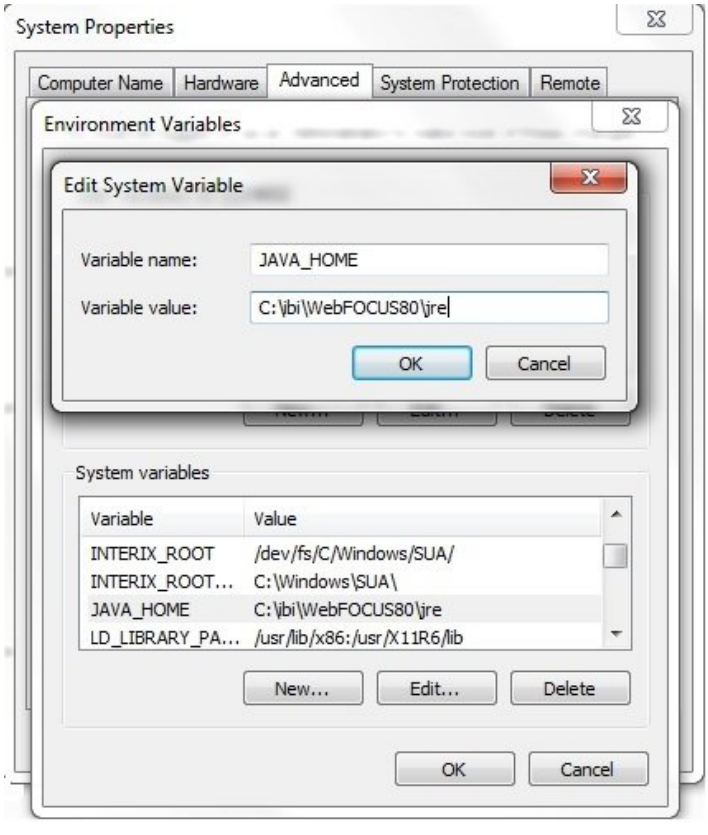
Configuring the Words Analysis Adapter

This section describes how to configure the Words Analysis Adapter.

As a prerequisite for configuring the Words Analysis Adapter, the path for the Java JDK or Java Runtime must be set. The WebFOCUS Reporting Server searches for the following variable names:

- **JDK_HOME.** Used to define the path for the Java JDK.
- **JAVA_HOME.** Used to define the path for the Java Runtime.

The following image shows how to set the JAVA_HOME variable on a Windows platform using the System Properties dialog.



Procedure: How to Configure the Words Analysis Adapter

1. From the Web Console Applications page, click *Get Data*.

or

From the Data Management Console, expand the *Adapters* folder.

In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.
3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.
4. Right-click *Words Analysis* and select *Configure*.

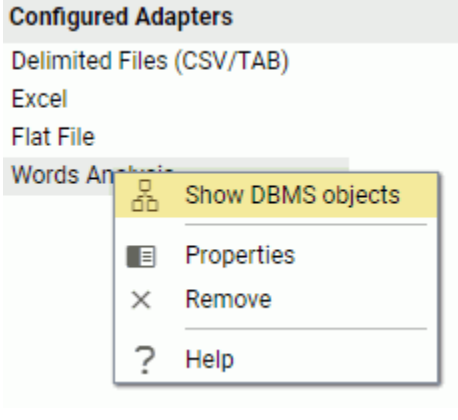
   The configured Words Analysis Adapter is added in the left pane.

Creating Metadata and Sample Reports for the Words Analysis Adapter

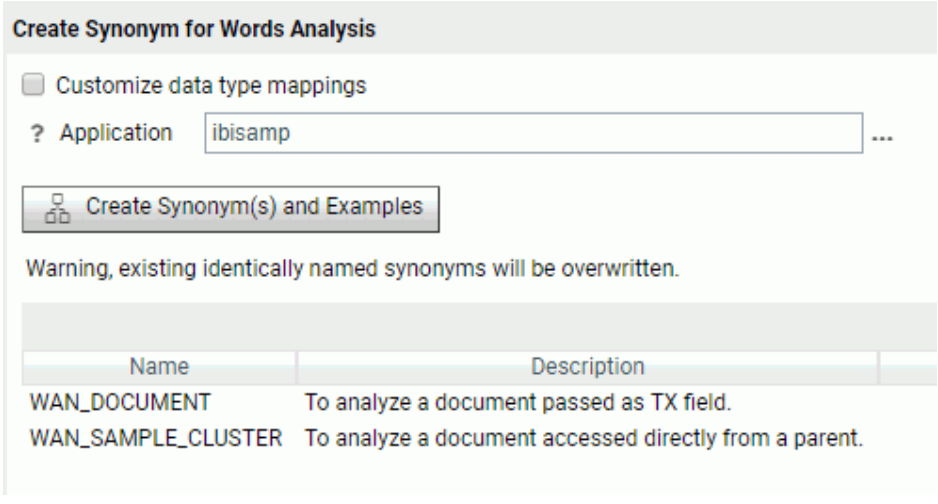
Create Synonym for the Words Analysis Adapter creates the metadata used for WebFOCUS reporting. It also creates sample WebFOCUS reports.

Procedure: How to Create Metadata and Sample Reports

1. From the WebFOCUS Reporting Server Web Console, expand the *Adapters* folder, and then the *Configured* folder.
2. Right-click *Words Analysis* and select *Show DBMS objects* from the context menu, as shown in the following image.



The Create Synonym for Words Analysis pane opens, as shown in the following image.



3. Enter a specific application in the Application field or click the ellipsis button to the right of the field to select an application where the metadata and sample reports are to be stored.
4. Click *Create Synonym(s) and Examples*.

   The Create Synonym for Words Analysis Status pane opens and indicates that the synonym was created successfully.

Words Analysis Adapter Examples

This section describes the metadata and sample reports for the Words Analysis Adapter.

Reference: Words Analysis Adapter Metadata

The following table lists and describes the available metadata for the Words Analysis Adapter.

Metadata	Description

wan_document	Used to pass textual data to the Words Analysis Adapter and return a count of occurrences for each word.
	
	By default, special characters (such as #,%,$,@) are excluded from the analysis. To include individual special characters in the analysis, they must be passed to the SYMBOLS field in the form of a selection. For example:
	
	`WHERE SYMBOLS EQ '#@'`
	
	Words contained in the Stopwords file are excluded from the analysis.
	The Access File (.acx) contains an attribute for the Stopwords file location:
	
	`STOPWORDS_FILENAME='wordsanalysis/wan_stopwords_ibi.txt'`
	
	The wan_stopwords_ibi.txt file is the default Stopwords file that gets loaded as part of Create Synonym.

wansampl/wan_sample_cluster	Cluster Join between *wansampl/wan_sample_fix* and *wan_document*.
wansampl/wan_sample_fix	Metadata that defines the sample text file (wansampl/wan_sample.txt) used for the *wan_sample_join*, *wan_sample_join_tagcloud*, and *wan_sample_cluster* sample reports.

Reference: Words Analysis Adapter Sample Reports

The following table lists and describes the sample reports for the Words Analysis Adapter.

Sample Report	Description
wansampl/wan_sample_cluster	Performs a count of the occurrences of words from the text passed in the *wansampl/wan_sample.txt* file using the Cluster Join master *wansampl/wan_sample_cluster*.
wansampl/wan_sample_join	Performs a count of the occurrences of words from the text passed in the *wansampl/wan_sample.txt* file. Joins *wansampl/wan_sample_fix* to *wan_document*.
wansampl/wan_sample_join_tagcloud	Performs a count of the occurrences of words from the text passed in the *wansampl/wan_sample.txt* file. The results are displayed in a tag cloud graph. Joins *wansampl/wan_sample_fix* to *wan_document*.
wansampl/wan_sample_where	Performs a count of the occurrences of words from the text passed in a WHERE statement. Also includes a WHERE statement defining the Stopword file(s) to be used in the request. It overrides the definition in the wan_document Access File (.acx). Uses: wan_document

Chapter 110

Using the Adapter for XML

The Adapter for XML allows applications to access XML data sources. The adapter traverses the hierarchical trees and returns an answer set to the requesting application.

In this chapter:

Preparing the XML Environment

The Adapter for XML does not require any environment variables to be set up.

Configuring the Adapter for XML

You can configure the Adapter for XML from the Web Console or the Data Management Console.

Procedure: How to Configure an Adapter

1. From the Web Console Applications page, click *Get Data*.

   or

   From the Data Management Console, expand the *Adapters* folder.

   In the DMC, the Adapters folder opens. In the Web Console, the Get Data page opens showing your configured adapters.

2. In the Web Console, click the *(+)* button, and find the adapter on the page or, in the DMC, expand the *Available* folder if it is not already expanded.
   On the Web Console, you can select a category of adapter from the drop-down list or use the search option (magnifying glass) to search for specific characters.

3. In the DMC, expand the appropriate group folder and the specific adapter folder. The group folder is described in the connection attributes reference.

4. Right-click the adapter name and/or version and select *Configure*.

   The Add Adapter to Configuration pane opens.

5. Enter values for the parameters required by the adapter, as described in the chapter for the specific adapter you want to configure.
6. Click *Configure*. The configured adapter is added to the Adapters list in the DMC resources tree or the Configured list in the Web Console.
   In the Web Console, the adapter remains on the Available Adapters list with an asterisk to indicate that at least one connection has been configured. You can configure additional connections from either the Configured or Available list by right-clicking the adapter and clicking *Add Connection*.

Managing XML Metadata

When the server accesses a data source, it needs to know how to interpret the data that it finds. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the XML data types.

Creating Synonyms

Synonyms define unique names (or aliases) for each XML data structure that is accessible from a server. Synonyms are useful because they hide the location and identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while enabling client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and Access File based on a given XML document.

Procedure: How to Create a Synonym

1. From the Web Console *Applications* page, click *Get Data*.
2. On the Configured Adapters section of the page, in Simple Mode, right-click an adapter and click *Show Connections*. Right-click a connection.

   Depending on the type of adapter you chose, one of the following options appears on the context menu.

   - **Show DBMS objects.** This opens the page for selecting synonym objects and properties.
   - **Create metadata objects.** This opens the page for selecting synonym objects and properties.
   - **Show files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.

- **Show local files.** This opens a file picker. After you choose a file of the correct type, the page for selecting synonym objects and properties opens.
- **Show topics.** This opens the page for selecting synonym objects and properties for topics within the Kafka environment.

3. Enter values for the parameters required by the adapter as described in the chapter for your adapter.
4. After entering the parameter values, click *Add*.

   The button may be labeled *Next*, *Create Synonym*, *Create Base Synonyms*, *Create Cluster Synonym*, or *Update Base Synonyms*.

   The synonym creation process for most adapters has been consolidated so that you can enter all necessary parameters on one page. However, for some adapters such as LDAP, you must click *Next* buttons until you get to a page that has a *Create Synonym* button.

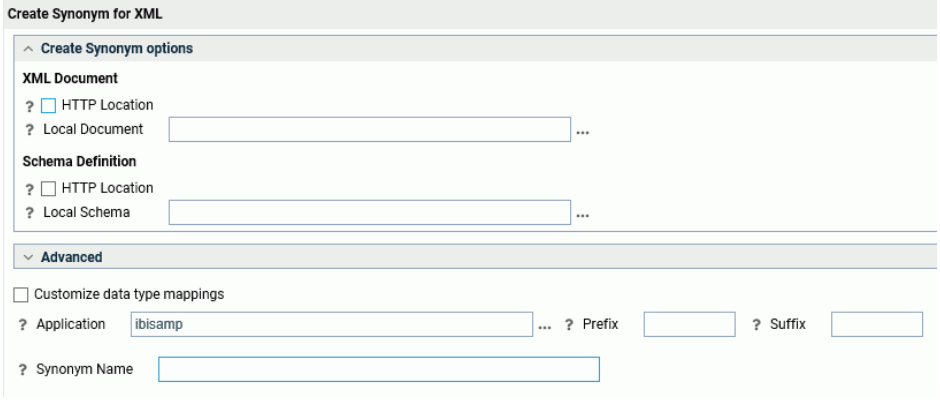
The synonym is created and added under the specified application directory.

Note: When creating a synonym, if you choose the Validate check box (where available), the server adjusts special characters and checks for reserved words. For more information, see *Validation for Special Characters and Reserved Words* on page 2743.

Reference: Synonym Creation Parameters for XML

The following list describes the parameters for which you will need to supply values, and related tasks you will need to complete in order to create a synonym for the adapter. These options may appear on multiple panes. To advance from pane to pane, click the buttons provided, ending with the *Create Synonym* button, which generates the synonym based on your entries.

You can create a synonym based on either an XML document or an XML schema, which can either be on your local file system or at a URL, as shown in the following image:



- If you want to base the synonym on an XML document, enter the required parameters in the *XML Document* section.
- If you want to base the synonym on an XML schema, enter the required parameters in the *Schema Definition* section..

HTTP Location

Enables you to select a document instance from a URL. This selection requires a Base Location, Document Name, and Document Extension. These fields become available if you select the *HTTP Location* check box. Enter the http address of a directory that contains the XML document you are using to create the synonym. (This functionality is not available when the XML document is a local file.) The URL must start with http:// or https://.

Local Document

Defines the location of the document instance. Enter a physical path or application directory and the XML document name, or click the ellipsis (...) to navigate to the document.

Document Name

Enter the name of the XML document.

Document Extension

Enter the document extension. The default is xml.

Advanced Section

Synonym field names processing options

Validate

Select the *Validate* check box if you wish to convert all special characters to underscores and perform a name check to prevent the use of reserved names. (This is accomplished by adding numbers to the names.) This parameter ensures that names adhere to specifications. See *Validation for Special Characters and Reserved Words* on page 2743 for more information.

When the Validate option is cleared, only the following characters are converted to underscores: '-'; ' '; ' \'; '/'; ','; '$'. No checking is performed for names.

Make unique

Select the *Make unique* check box if you wish to set the scope for field and group names to the entire synonym. This ensures that no duplicate names are used, even in different segments of the synonym. When this option is unchecked, the scope is the segment.

Synonym Name

Indicates the name that will be assigned to the synonym. To assign a different name, replace the displayed value.

Write

When using a schema file (.xsd), this option includes more information in the synonym Access File. The additional information reflects CHOICE and SEQUENCE definitions of complex XML data types and is used in XML MODIFY.

Position

Defines the XPATH to the element subtree on which the synonym will be based.

Application

Select an application directory. The default value is baseapp.

Prefix/Suffix

If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.

If all tables and views have unique names, leave the prefix and suffix fields blank.

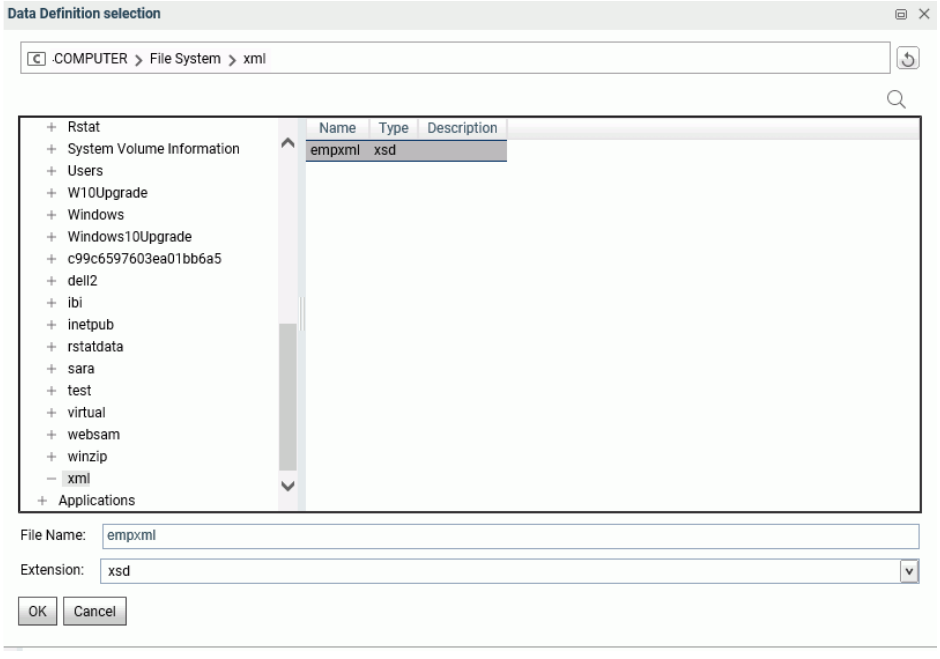
Once you have entered the parameters, click *Create Synonym* on the ribbon.

Example: Creating a Synonym for the empxml xsd

empxml.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://
example.org/employee/" targetNamespace="http://example.org/employee/">
    <xsd:simpleType name="genderIdentifiers">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="male"/>
            <xsd:enumeration value="female"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="socialSecurityNumber">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="\d{3}\-\d{2}\-\d{4}"/>
                </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="annotatedAge">
        <xsd:simpleContent>
            <xsd:extension base="xsd:unsignedShort"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="Person">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="sex" type="tns:genderIdentifiers"/>
            <xsd:element name="age" type="tns:annotatedAge"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Employee">
        <xsd:complexContent>
            <xsd:extension base="tns:Person">
                <xsd:sequence>
                    <xsd:element name="ssnum"
type="tns:socialSecurityNumber"/>
                    <xsd:element name="salary" type="xsd:double"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="employee" type="tns:Employee"/>
</xsd:schema>
```

To generate a synonym from the empxml xsd, enter the location of the xsd file on the Create Synonym pane of the Web Console or the Data Management Console or click the ellipsis to navigate to the xsd file on your local file system, as shown in the following image.



In this case, the .xsd file is on the file system at:

c:\xml\empxml.xsd

1. Click *Next*.
2. Select *empxml* in the *Name* field.
3. Click *Create Synonym* on the ribbon. The synonym is created and added under the specified application directory (ibisamp is the default).
4. Open the *ibisamp* application folder in the navigation pane, right-click the *empxml* synonym, and choose *Edit as Text* to view the generated Master File.
5. Right-click the *empxml* synonym and choose *Edit Access File as Text* to view the corresponding Access File.

Generated Master File empxml.mas

```
FILENAME=EMPXML, SUFFIX=XML     , $
  SEGMENT=EMPLOYEE, SEGTYPE=S0, $
    FIELDNAME=EMPLOYEE, ALIAS=employee, USAGE=A1, ACTUAL=A1,
ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=NAME, ALIAS=name, USAGE=A10, ACTUAL=A10,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=SEX, ALIAS=sex, USAGE=A10, ACTUAL=A10,
      ACCEPT='male' OR 'female',
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=AGE, ALIAS=age, USAGE=I5, ACTUAL=A5,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=SSNUM, ALIAS=ssnum, USAGE=A10, ACTUAL=A10,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=SALARY, ALIAS=salary, USAGE=E24.16, ACTUAL=A24,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
```

Generated Access File empxml.acx

```
SEGNAME=EMPLOYEE,
   TARGETNS=http://example.org/employee/,
   LOCATION=C:\xml\empxml.xsd, $
  ID=tns,
    NS=http://example.org/employee/,
    ELEMFORM=unqualified,
    ATTRFORM=unqualified, $
   FIELD=EMPLOYEE,
     TYPE=Employee,
     NS_ID=tns, $
   FIELD=SEX,
     TYPE=genderIdentifiers,
     NS_ID=tns, $
   FIELD=AGE,
     TYPE=annotatedAge,
     NS_ID=tns, $
   FIELD=SSNUM,
     TYPE=socialSecurityNumber,
     NS_ID=tns, $
```

Syntax: How to Create Synonyms From Schema Subsets

If you want to create a synonym from a large XML schema, you can improve performance by using a subset based on a given position in the hierarchy. This creates a smaller synonym from a subtree (child element) of the XML root.

The syntax is:

```
CREATE SYNONYM child_element FOR schema.xsd
DBMS XML
AT app_dir
PARMS POSITION=/parent_element/child_element
DROP
```

```
END
```

where:

child_element

Is the subtree from which the synonym is created.

schema.xsd

Is the schema on which the synonym is based.

app_dir

Is the application directory where the synonym is created.

PARMS POSITION

Indicates the position in the schema hierarchy.

parent_element

Is the parent element located at the top-level of the schema hierarchy.

In the following example, the synonym is created for the chapter subtree. The chapter is a child of the book element, which is located at the top-level of the xmlbook schema hierarchy:

```
CREATE SYNONYM chapter FOR xmlbook.xsd
DBMS XML
AT baseapp
PARMS POSITION=/book/chapter
DROP
END
```

XPATH=/book is added to the Access File to indicate the position of chapter (the root segment in the synonym) in the schema hierarchy.

When issuing TABLE requests or writing XML based on the synonym, the actual path of the node is required. Since chapter is not the root node in the original schema, the path from the top of the schema to chapter is /book/chapter.

When writing XML files based on the synonym, the chapter element will be wrapped inside the book element, as in <book><chapter>...</chapter></book>.

When reporting on an XML file based on the synonym, you should overlook book since chapter is right under book.

Reference: Managing Synonyms

Once you have created a synonym, you can right-click the synonym name in the Adapter navigation pane of either the Web Console or the Data Management Console to access the available options.

For a list of options, see *Synonym Management Options* on page 93

Accessing XML Documents From a Relational DBMS XML Data Type

XML documents might be stored in any fields or columns in any data source. Reporting from such documents is supported by defining their structure as subtrees attached to a parent segment which describes the original data.

The synonym creation process must be run against the data in the DBMS and against the XML document. The two Master Files must then be combined to make the XML Master File a child of the Master File created against the DBMS. A FILEDEF is not needed in this instance.

Procedure: How to Access XML Data From an RDBMS Using Web Console or Data Management Console Tools

1. Using the Web Console or the Data Management Console Create Synonym facility, generate a synonym for an RDBMS data source that contains a column of XML data. Regardless of the data type used to contain the XML data in the native data source, it will be mapped as a TX column in the Master File synonym. (For example, for Db2 Version 9, the XML data type is mapped to TX; for many RDBMS, the CLOB data type is mapped to TX.)
2. Open the generated Master File in the Synonym Editor. The Master File appears in the right pane in Text View. For example, the Master File for a Progress data source might look as follows:

   ```
   FILE=XMLPRO1 ,SUFFIX=SQLPRO ,$
   SEGNAME=XMLPRO1 ,SEGTYPE=S0 ,$
   FIELD=FLD1 ,FLD1 ,A2   ,A2 ,MISSING=ON ,$
   FIELD=FLD2 ,FLD2 ,TX50 ,TX ,MISSING=ON ,$
   FIELD=FLD3 ,FLD3 ,TX50 ,TX ,MISSING=ON ,$
   ```

   Notice that this example has two TX columns, each of which contains different data and requires a separate segment declaration in the Master File.
3. From the DMC, click a column described as TX. The pop-up menu contains the Map External XML option.

   The *Map External XML* option reads the XML data directly and creates the structure. The resulting Master File contains the definition of the XML data, represented as a new segment called SEGSUF=XML, which appears in the Text View pane following the original RDBMS segment. The Master File might look like the following:

```
FILE=XMLPRO1    ,SUFFIX=SQLPRO ,$
SEGNAME=XMLPRO1 ,SEGTYPE=S0    ,$
FIELD=FLD1 ,FLD1 ,A2   ,A2  ,MISSING=ON ,$
FIELD=FLD2 ,FLD2 ,TX50 ,TX  ,MISSING=ON ,$
FIELD=FLD3 ,FLD33,TX50 ,TX  ,MISSING=ON ,$
SEGMENT=ORDER, SEGTYPE=S0,POSITION=FLD2,PARENT=XMLPRO1,SEGSUF=XML, $
  FIELDNAME=ORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATUS, ALIAS='Status', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=TOTALPRICE, ALIAS='TotalPrice', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DATE, ALIAS='Date', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=PRIORITY, ALIAS='Priority', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CLERK, ALIAS='Clerk', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SHIPPRIORITY,ALIAS='ShipPriority',USAGE=A10,ACTUAL=A10,$
  FIELDNAME=COMMENT, ALIAS='Comment', USAGE=A10, ACTUAL=A10, $
SEGMENT=PORDER, SEGTYPE=S0,POSITION=FLD3,PARENT=XMLPRO1,SEGSUF=XML $
    FIELDNAME=PORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
    FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=ADDRESS, ALIAS='Address', USAGE=A10, ACTUAL=A10, $
    FIELDNAME=STATE, ALIAS='State', USAGE=A10, ACTUAL=A10, $
```

Tip: If you have a TX column that contains multiple XML formats (for example, a name field and an address field), you can choose *Map External XML* multiple times to create a separate SEGSUF=XML segment for each format.

4. From the Synonym Editor's File menu, save the updated Master File.

Procedure: How to Access XML Data From an RDBMS Manually

Suppose that you have a table in an RDBMS with one or more columns storing XML data. In order to report from the XML data, following these steps:

1. Create the Master File for the relational data source using the format for that DBMS.

```
FILE=XMLPRO1 ,SUFFIX=SQLPRO ,$
SEGNAME=XMLPRO1 ,SEGTYPE=S0 ,$
FIELD=FLD1 ,FLD1 ,A2   ,A2 ,MISSING=ON ,$
FIELD=FLD2 ,FLD2 ,TX50 ,TX ,MISSING=ON ,$
FIELD=FLD3 ,FLD3 ,TX50 ,TX ,MISSING=ON ,$
```

2. Create a Master File for the XML document in the column of the RDBMS table. If there are two XML documents with different formats, you must create a Master File for each one.
3. Manually combine the Master Files. On each root segment for the XML Master File, add three fields: position, parent and segsuf. The POSITION keyword identifies the field containing the XML document. The PARENT field describes the original data source. The field SEGSUF defines the root segment of an XML document representing sub-tree. The total length of all fields in the Master File must not exceed the FOCUS limitation of 32k. If it does, the query will fail.

```
FILENAME=BASEAPP/ORDER, SUFFIX=XML    , $
 SEGMENT=ORDER, SEGTYPE=S0, $
  FIELDNAME=ORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATUS, ALIAS='Status', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=TOTALPRICE, ALIAS='TotalPrice', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DATE, ALIAS='Date', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=PRIORITY, ALIAS='Priority', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CLERK, ALIAS='Clerk', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SHIPPRIORITY,ALIAS='ShipPriority',USAGE=A10,ACTUAL=A10,$
  FIELDNAME=COMMENT, ALIAS='Comment', USAGE=A10, ACTUAL=A10, $
FILENAME=BASEAPP/PORDER, SUFFIX=XML     , $
 SEGMENT=PORDER, SEGTYPE=S0, $
  FIELDNAME=PORDER, ALIAS='POrder', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=ADDRESS, ALIAS='Address', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATE, ALIAS='State', USAGE=A10, ACTUAL=A10, $
```

Combined Master file:

```
FILE=XMLPRO1    ,SUFFIX=SQLPRO ,$
SEGNAME=XMLPRO1 ,SEGTYPE=S0    ,$
FIELD=FLD1 ,FLD1 ,A2   ,A2  ,MISSING=ON ,$
FIELD=FLD2 ,FLD2 ,TX50 ,TX  ,MISSING=ON ,$
FIELD=FLD3 ,FLD33,TX50 ,TX  ,MISSING=ON ,$
SEGMENT=ORDER, SEGTYPE=S0,POSITION=FLD2,PARENT=XMLPRO1,SEGSUF=XML, $
  FIELDNAME=ORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATUS, ALIAS='Status', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=TOTALPRICE, ALIAS='TotalPrice', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=DATE, ALIAS='Date', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=PRIORITY, ALIAS='Priority', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CLERK, ALIAS='Clerk', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=SHIPPRIORITY,ALIAS='ShipPriority',USAGE=A10,ACTUAL=A10,$
  FIELDNAME=COMMENT, ALIAS='Comment', USAGE=A10, ACTUAL=A10, $
SEGMENT=PORDER, SEGTYPE=S0,POSITION=FLD3,PARENT=XMLPRO1,SEGSUF=XML $
  FIELDNAME=PORDER, ALIAS='Order', USAGE=A1, ACTUAL=A1, $
  FIELDNAME=KEY, ALIAS='key', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=CUSTOMER, ALIAS='Customer', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=ADDRESS, ALIAS='Address', USAGE=A10, ACTUAL=A10, $
  FIELDNAME=STATE, ALIAS='State', USAGE=A10, ACTUAL=A10, $
```

Using Static Joins

You can describe various views of the same physical XML document using Master Files and Access Files.

- In the Master File, you use REFERENCE attributes in field definitions to reflect physical relationships between tags in XML document hierarchies and PARENT attributes, which establish logical hierarchical relationships.
- In the Access File, you use the KEYFLD and IXFLD attributes to identify the XML tags that act as primary/foreign keys in the XML document hierarchies, which establish the logical join relationships. The parent field (foreign key) defined in KEYFLD supplies the value for cross-referencing. The descendant field (primary key) defined in IXFLD contains the corresponding value.

The adapter implements join matching values at run time.

You can use static Joins to create join relationships between hierarchically unrelated integral schema ComplexType definitions using any combination of data nodes.

Any XML tags belonging to these definitions can be used to create join pairs. In addition, you can multiply instances of the same physical segment to reflect logical join relationships as needed.

Note: Static (embedded) Joins are not directly supported by the Create Synonym facility. To take advantage of this feature, after a synonym is generated you must modify its Master and Access Files. The Data Management Console Synonym Editor is the recommended tool for such modifications since it provides an easily identifiable segments hierarchy with drag and drop capability and a visual calculator for KEYFLD/IXFLD modifications. Using the Data Management Console, you can quickly add all duplicate segments to the Master File, then delete any unneeded segments.

In some cases, schemas (such as those produced by Microsoft tools and reflecting relationships between the original MS SQL Server database tables) contain proper XML constraints (unique/key/keyref) for describing joins. You can use this information to modify the Master File:

- The elements *unique* and *key* define the primary key for the element (segment).
- The element *keyref* defines the *foreign* key for the element (segment).

For an illustration of such a schema, see *Modifying Master and Access Files Produced From Schema Exported by an SQL Server* on page 2670.

Example: Modifying Master and Access Files Produced From Schema Exported by an SQL Server

Suppose that two links exist in the schema defined in the file GetAllEmployeesFullDS.xml between the elements Employee and EmployeeTL (segments EMPLOYEE and EMPLOYEETL in the Master File). Two variations follow:

Defines EmployeeTL –> Employee joined by EId/EId pair

In this example, segment EMPLOYEE becomes the child of segment EMPLOYEETL in the Master File, and the Access File describes the foreign and primary keys as follows:

```
<xs:unique name="Constraint1" msdata:PrimaryKey="true">
   <xs:selector xpath=".//Employee" />
   <xs:field xpath="EId" />
</xs:unique>

<xs:keyref name="EmployeeTl_Employee_EId_EId" refer="Constraint1"
msdata:ConstraintOnly="true">
   <xs:selector xpath=".//EmployeeTl" />
   <xs:field xpath="EId" />
</xs:keyref>
```

Access File

```
SEGNAME=EMPLOYEE, KEYFLD=EID, IXFLD=EID, $
```

Defines Employee –> EmployeeTL joined relationship by TLInformation/EtId pair

In this example, segment EMPLOYEETL becomes the child of segment EMPLOYEE in the Master File, and the Access File describes the foreign and primary keys as follows:

```
<xs:unique name="EmployeeTl_Constraint1"
msdata:ConstraintName="Constraint1" msdata:PrimaryKey="true">
   <xs:selector xpath=".//EmployeeTl" />
   <xs:field xpath="EtId" />
</xs:unique>

<xs:keyref name="Employee_EmployeeTl_EtId_TLInformation"
refer="EmployeeTl_Constraint1" msdata:ConstraintOnly="true">
   <xs:selector xpath=".//Employee" />
   <xs:field xpath="TLInformation" />
</xs:keyref>
```

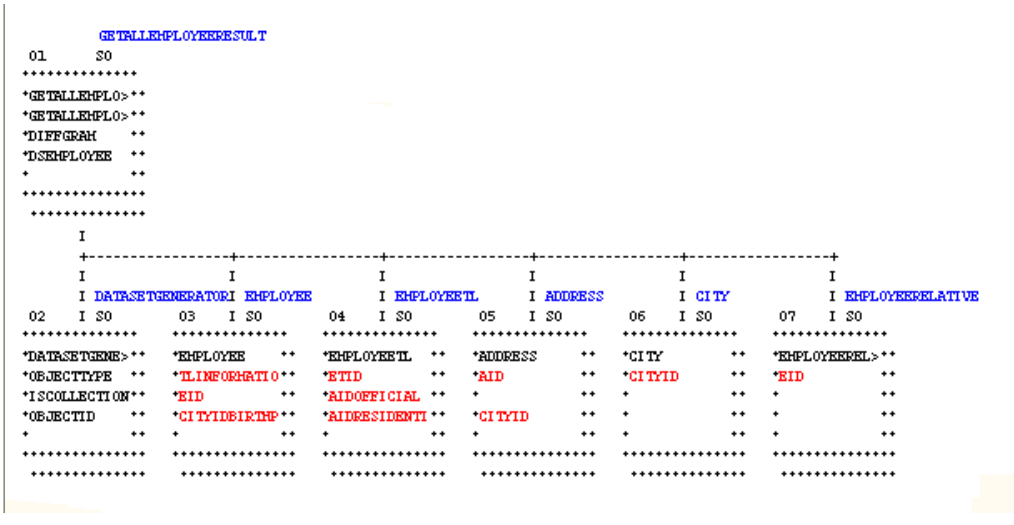
Access File

```
SEGNAME=EMPLOYEETL, KEYFLD=TLINFORMATION, IXFLD=ETID, $
```

Example: **Sample Join Structures: Original and Modified**

This example illustrates an original structure without a static join and the modifications made to reflect the use of static Joins.

Original Structure Without Static Joins



In this native structure all child segments have a common parent; the root segment GETALLEMPLOYEERESULTS.

In the corresponding Master File generated as part of the synonym, the PARENT attribute of each segment declaration describes the logical relationships among segments, while the REFERENCE attribute for each field provides information about the physical parent/child relationships among the elements in the file.

Generated Master File

```
FILENAME=GETALLEMPLOYEEFULLDS, SUFFIX=XML,
  DATASET=C:\Users\yn05149\apps\xml\GetAllEmployeesFullDS.xml, $
$
```

```
  SEGMENT=GETALLEMPLOYEESRESULT, SEGTYPE=S0, $
    FIELDNAME=GETALLEMPLOYEESRESULT, ALIAS=GetAllEmployeesResponse, USAGE=A1,
      ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=GETALLEMPLOYEESRESULT1, ALIAS=GetAllEmployeesResult, USAGE=A1,
      ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GETALLEMPLOYEESRESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=DIFFGRAM, ALIAS=diffgram, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GETALLEMPLOYEESRESULT1, PROPERTY=ELEMENT,  $
    FIELDNAME=DSEMPLOYEE, ALIAS=DSEmployee, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=DIFFGRAM, PROPERTY=ELEMENT,  $
$

  SEGMENT=DATASETGENERATOR, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=DATASETGENERATOR, ALIAS=DataSetGenerator, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=DSEMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=OBJECTTYPE, ALIAS=ObjectType, USAGE=A30, ACTUAL=A30,
      REFERENCE=DATASETGENERATOR, PROPERTY=ELEMENT,  $
    FIELDNAME=ISCOLLECTION, ALIAS=IsCollection, USAGE=A5, ACTUAL=A5,
      REFERENCE=DATASETGENERATOR, PROPERTY=ELEMENT,  $
    FIELDNAME=OBJECTID, ALIAS=ObjectId, USAGE=I11, ACTUAL=A11,
      REFERENCE=DATASETGENERATOR, PROPERTY=ELEMENT,  $
$

  SEGMENT=EMPLOYEE, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=EMPLOYEE, ALIAS=Employee, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=DSEMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=EID, ALIAS=EId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=CITYIDBIRTHPLACE, ALIAS=CityIdBirthplace, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=TLID, ALIAS=TlId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=EFIRSTNAME, ALIAS=EFirstname, USAGE=A30, ACTUAL=A30,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=ENAME, ALIAS=EName, USAGE=A30, ACTUAL=A30,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=EBIRTHDATE, ALIAS=EBirthdate, USAGE=HYYMDm, ACTUAL=A35,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=IEID, ALIAS=IeId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=TLINFORMATION, ALIAS=TLInformation, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
$
```

```
  SEGMENT=EMPLOYEETL, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=EMPLOYEETL, ALIAS=EmployeeTl, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=ETID, ALIAS=EtId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=EID, ALIAS=EId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=AIDOFFICIAL, ALIAS=AIdOfficial, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=AIDRESIDENTIAL, ALIAS=AIdResidential, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=DEPID, ALIAS=DepId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
$

  SEGNAME=ADDRESS, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=ADDRESS, ALIAS=Address, USAGE=A1, ACTUAL=A1,
      PROPERTY=ELEMENT, REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE,
      ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=AID, ALIAS=AId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=ADDRESS, $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=ADDRESS, $
    FIELDNAME=AADDRESS, ALIAS=AAddress, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=ADDRESS, $
    FIELDNAME=AADDRESSNMBR, ALIAS=AAddressNmbr, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=ADDRESS, $
    FIELDNAME=ATEL1, ALIAS=ATel1, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=ADDRESS, $
    FIELDNAME=ATEL2, ALIAS=ATel2, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=ADDRESS, $
$

  SEGNAME=CITY, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=CITY, ALIAS=City, USAGE=A1, ACTUAL=A1, PROPERTY=ELEMENT,
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=CITY, $
    FIELDNAME=COUNTRYID, ALIAS=CountryId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=CITY, $
    FIELDNAME=CITYNAME, ALIAS=CityName, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=CITY, $
    FIELDNAME=CITYPOSTALCODE, ALIAS=CityPostalcode, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=CITY, $
$
```

```
  SEGNAME=EMPLOYEERELATIVE, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=EMPLOYEERELATIVE, ALIAS=EmployeeRelative, USAGE=A1, ACTUAL=A1,
      PROPERTY=ELEMENT,
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=ERID, ALIAS=ErId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
    FIELDNAME=EID, ALIAS=EId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
    FIELDNAME=CITYIDBIRTHPLACE, ALIAS=CityIdBirthplace, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
    FIELDNAME=TLID, ALIAS=TlId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
    FIELDNAME=ERLRELATIONSHIP, ALIAS=ErLRelationship, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
    FIELDNAME=ERNAME, ALIAS=ErName, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
    FIELDNAME=ERFIRSTNAME, ALIAS=ErFirstname, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE, $
$
```

Generated Access File. When you create a synonym, the generated Access File stores entity abbreviations but does not contain other content. However, you can open and edit this file to make any required modifications, as will be illustrated shortly in the structure that defines static Joins.

Modified Structure With Static Joins

In the modified structure, the original two-level organization is replaced by a five-level structure in which some instances of the same physical segments (CITY and ADDRESS) have been multiplied to reflect the new logical Join relationships.

Modified Master File

In the corresponding Master File, parent/child relationships have been *manually* expanded and rearranged and references have been revised to provide accurate pointers through the new hierarchy.

As you look at this Master File, keep the following information in mind:

- The REFERENCE attributes in the field definitions reflect the physical relationships between elements in XML document hierarchies and the segment level PARENT attributes, which establish logical hierarchical relationships. Notice that for segments that have been added or whose positions in the hierarchy have changed, the REFERENCE attribute in the Master File now generally consists of two parts:

  segment.fieldname

  As a rule, the adapter attempts to resolve each reference in the current segment. If that cannot be accomplished, it searches for the first occurrence of the REFERENCE in the parent chain. (This is illustrated by the REFERENCE attributes in SEGMENT EMPLOYEETL.) Qualified (two-part) references force the direct resolution in the specified segment. (This is illustrated by the REFERENCE attributes in SEGMENT EMPLOYEERELATIVE.) Both notations are correct and produce the same result in this example.

- The ALIAS attribute in the Master File maps to the element name in the native xml schema so it cannot be changed. However, since the FIELDNAME is always associated with the ALIAS, the field name can be changed without jeopardizing the reference.

- The REFERENCE attribute points to the field name, hence, if field name is changed all references to it should be changed accordingly.

```
FILENAME=GETALLEMPLOYEEFULLDS, SUFFIX=XML,
  DATASET=C:\Users\yn05149\apps\xml\GetAllEmployeesFullDS.xml, $
$

 SEGMENT=GETALLEMPLOYEESRESULT, SEGTYPE=S0, $
    FIELDNAME=GETALLEMPLOYEESRESULT, ALIAS=GetAllEmployeesResponse, USAGE=A1,
      ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=GETALLEMPLOYEESRESULT1, ALIAS=GetAllEmployeesResult, USAGE=A1,
      ACTUAL=A1, ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GETALLEMPLOYEESRESULT, PROPERTY=ELEMENT,  $
    FIELDNAME=DIFFGRAM, ALIAS=diffgram, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GETALLEMPLOYEESRESULT1, PROPERTY=ELEMENT,  $
    FIELDNAME=DSEMPLOYEE, ALIAS=DSEmployee, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=DIFFGRAM, PROPERTY=ELEMENT,  $
$
```

```
 SEGMENT=DATASETGENERATOR, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=DATASETGENERATOR, ALIAS=DataSetGenerator, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=DSEMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=OBJECTTYPE, ALIAS=ObjectType, USAGE=A30, ACTUAL=A30,
      REFERENCE=DATASETGENERATOR, PROPERTY=ELEMENT,  $
    FIELDNAME=ISCOLLECTION, ALIAS=IsCollection, USAGE=A5, ACTUAL=A5,
      REFERENCE=DATASETGENERATOR, PROPERTY=ELEMENT,  $
    FIELDNAME=OBJECTID, ALIAS=ObjectId, USAGE=I11, ACTUAL=A11,
      REFERENCE=DATASETGENERATOR, PROPERTY=ELEMENT,  $
$

 SEGMENT=EMPLOYEE, SEGTYPE=S0, PARENT=GETALLEMPLOYEESRESULT, $
    FIELDNAME=EMPLOYEE, ALIAS=Employee, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=DSEMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=EID, ALIAS=EId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=CITYIDBIRTHPLACE, ALIAS=CityIdBirthplace, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=TLID, ALIAS=TlId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=EFIRSTNAME, ALIAS=EFirstname, USAGE=A30, ACTUAL=A30,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=ENAME, ALIAS=EName, USAGE=A30, ACTUAL=A30,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=EBIRTHDATE, ALIAS=EBirthdate, USAGE=HYYMDm, ACTUAL=A35,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=IEID, ALIAS=IeId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=TLINFORMATION, ALIAS=TLInformation, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEE, PROPERTY=ELEMENT,  $
$

  SEGMENT=EMPLOYEETL, SEGTYPE=S0, PARENT=EMPLOYEE, $
    FIELDNAME=EMPLOYEETL, ALIAS=EmployeeTl, USAGE=A1, ACTUAL=A1,
      ACCESS_PROPERTY=(INTERNAL),
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, PROPERTY=ELEMENT,  $
    FIELDNAME=ETID, ALIAS=EtId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=EID, ALIAS=EId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=AIDOFFICIAL, ALIAS=AIdOfficial, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=AIDRESIDENTIAL, ALIAS=AIdResidential, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
    FIELDNAME=DEPID, ALIAS=DepId, USAGE=I11, ACTUAL=A11,
      REFERENCE=EMPLOYEETL, PROPERTY=ELEMENT,  $
$
```

```
  SEGNAME=OADDRESS, SEGTYPE=S0, PARENT=EMPLOYEETL, $
    FIELDNAME=ADDRESS, ALIAS=Address, USAGE=A1, ACTUAL=A1,
      PROPERTY=ELEMENT, REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE,
      ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=AID, ALIAS=AId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
    FIELDNAME=AADDRESS, ALIAS=AAddress, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
    FIELDNAME=AADDRESSNMBR, ALIAS=AAddressNmbr, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
    FIELDNAME=ATEL1, ALIAS=ATel1, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
    FIELDNAME=ATEL2, ALIAS=ATel2, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
$

  SEGNAME=OCITY, SEGTYPE=S0, PARENT=OADDRESS, $
    FIELDNAME=CITY, ALIAS=City, USAGE=A1, ACTUAL=A1, PROPERTY=ELEMENT,
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=OCITY.CITY, $
    FIELDNAME=COUNTRYID, ALIAS=CountryId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=OCITY.CITY, $
    FIELDNAME=CITYNAME, ALIAS=CityName, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OCITY.CITY, $
    FIELDNAME=CITYPOSTALCODE, ALIAS=CityPostalcode, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OCITY.CITY, $
$

  SEGNAME=RADDRESS, SEGTYPE=S0, PARENT=EMPLOYEETL, $
    FIELDNAME=ADDRESS, ALIAS=Address, USAGE=A1, ACTUAL=A1,
      PROPERTY=ELEMENT, REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE,
      ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=AID, ALIAS=AId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=RADDRESS.ADDRESS, $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=RADDRESS.ADDRESS, $
    FIELDNAME=AADDRESS, ALIAS=AAddress, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=RADDRESS.ADDRESS, $
    FIELDNAME=AADDRESSNMBR, ALIAS=AAddressNmbr, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=RADDRESS.ADDRESS, $
    FIELDNAME=ATEL1, ALIAS=ATel1, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=RADDRESS.ADDRESS, $
    FIELDNAME=ATEL2, ALIAS=ATel2, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=OADDRESS.ADDRESS, $
$
```

```
  SEGNAME=RCITY, SEGTYPE=S0, PARENT=RADDRESS, $
    FIELDNAME=CITY, ALIAS=City, USAGE=A1, ACTUAL=A1, PROPERTY=ELEMENT,
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=RCITY.CITY, $
    FIELDNAME=COUNTRYID, ALIAS=CountryId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=RCITY.CITY, $
    FIELDNAME=CITYNAME, ALIAS=CityName, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=RCITY.CITY, $
    FIELDNAME=CITYPOSTALCODE, ALIAS=CityPostalcode, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=RCITY.CITY, $
$

  SEGNAME=BCITY, SEGTYPE=S0, PARENT=EMPLOYEE, $
    FIELDNAME=CITY, ALIAS=City, USAGE=A1, ACTUAL=A1, PROPERTY=ELEMENT,
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=CITYID, ALIAS=CityId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=BCITY.CITY, $
    FIELDNAME=COUNTRYID, ALIAS=CountryId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=BCITY.CITY, $
    FIELDNAME=CITYNAME, ALIAS=CityName, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=BCITY.CITY, $
    FIELDNAME=CITYPOSTALCODE, ALIAS=CityPostalcode, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=BCITY.CITY, $
$

  SEGNAME=EMPLOYEERELATIVE, SEGTYPE=S0, PARENT=EMPLOYEE, $
    FIELDNAME=EMPLOYEERELATIVE, ALIAS=EmployeeRelative, USAGE=A1, ACTUAL=A1,
      PROPERTY=ELEMENT,
      REFERENCE=GETALLEMPLOYEESRESULT.DSEMPLOYEE, ACCESS_PROPERTY=(INTERNAL), $
    FIELDNAME=ERID, ALIAS=ErId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
    FIELDNAME=EID, ALIAS=EId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
    FIELDNAME=CITYIDBIRTHPLACE, ALIAS=CityIdBirthplace, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
    FIELDNAME=TLID, ALIAS=TlId, USAGE=I11, ACTUAL=A11,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
    FIELDNAME=ERLRELATIONSHIP, ALIAS=ErLRelationship, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
    FIELDNAME=ERNAME, ALIAS=ErName, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
    FIELDNAME=ERFIRSTNAME, ALIAS=ErFirstname, USAGE=A30, ACTUAL=A30,
      PROPERTY=ELEMENT, REFERENCE=EMPLOYEERELATIVE.EMPLOYEERELATIVE, $
$
```

Modified Access File

The modified Access File shows which keys are used in the logical structure of the Join:

- The parent field (foreign key) defined in KEYFLD supplies the value for cross-referencing.
- The descendant field (primary key) defined in IXFLD contains the corresponding value.

```
SEGNAME=EMPLOYEETL,       KEYFLD=TLINFORMATION,    IXFLD=ETID,    $
SEGNAME=OADDRESS,         KEYFLD=AIDOFFICIAL,      IXFLD=AID,     $
SEGNAME=OCITY,            KEYFLD=CITYID,           IXFLD=CITYID,  $
SEGNAME=RADDRESS,         KEYFLD=AIDRESIDENTIAL,   IXFLD=AID,     $
SEGNAME=RCITY,            KEYFLD=CITYID,           IXFLD=CITYID,  $
SEGNAME=BCITY,            KEYFLD=CITYIDBIRTHPLACE, IXFLD=CITYID,  $
SEGNAME=EMPLOYEERELATIVE, KEYFLD=EID,              IXFLD=EID,     $
```

Data Type Support

The Create Synonym process uses an XSD or XML document as the source for creating the synonym.

If you try to perform arithmetic operations (-, +, >, <) on fields that are numbers or dates in the XML document but are mapped as ALPHA in the Master File, you may not get the expected results since the arithmetic operations are performed on string literals. To override this problem you may modify the USAGE attribute of a field as described in the following sections.

Note: The data type defined in the ACTUAL attribute of a field must remain as ALPHA.

In order to change the default setting you must issue the SET VARCHAR command.

The following table lists how the server maps XSD data types in a Master File. You can change some of these mapping defaults, as described in *Changing the Length of Character Strings* on page 2682 and *Changing the Precision and Scale of Numeric Columns* on page 2683.

XSD Data Type	USAGE Attribute	ACTUAL Attribute
decimal	P20.3	A20
integer	P33	A33
nonPositiveInteger	33	A33
negativeInteger	P33	A33
boolean	A5	A5
long	P20	A20
int	I11	A11
short	I6	A6
byte	I4	A4

XSD Data Type	USAGE Attribute	ACTUAL Attribute
nonNegativeInteger	P32	A32
unsignedLong	P20	A20
unsignedInt	P10	A10
unsignedShort	I5	A5
unsignedByte	I4	A4
positiveInteger	P32	A32
double	D20.2	A20
float	F15.2	A15
dateTime	HYYMDm	A27
time	HHISsm	A15
date	YYMD	A10
gYearMonth	HYYM	A8
gYear	HYY	A5
gMonthDay	HMD	A6
gDay	HD	A3
gMonth	HM	A4
string	A30	A30
normalizedString	A30	A30
token	A30	A30
Name	A30	A30
NMTOKEN	A30	A30
ID	A30	A30

XSD Data Type	USAGE Attribute	ACTUAL Attribute
hexBinary	A30	A30
language	A30	A30
anyURI	A30	A30
QName	A30	A30

Syntax: How to Set the Actual and Usage Attributes

```
ENGINE XML SET VARCHAR {ON|OFF}
```

The ACTUAL and USAGE attributes of each field in the Master File is set arbitrarily to A10. You can override this setting using the SET FIELDLENGTH command

```
ENGINE XML SET FIELDLENGTH nnn
```

where:

nnn

Is the length assigned to actual and usage attributes of all fields in the Master File during the create synonym process. The maximum length is 3000.

Changing the Length of Character Strings

By default, when the Adapter for XML creates a synonym, it maps all fields defined as string in the XML schema to 10-byte fixed-length alphanumeric in the Master File. You can change the length (the variability and the number of bytes) using the SET VARCHAR and SET LENGTH commands.

Syntax: How to Switch Between Variable and Fixed-length Strings

By default, when the Adapter for XML creates a synonym, it maps all fields defined as string in the XML schema to fixed-length alphanumeric in the Master File. You can change this default to variable length using the SET VARCHAR command

```
ENGINE XML SET VARCHAR {ON|OFF}
```

where:

`XML`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`ON`

Maps all fields defined as string in the XML schema to variable-length alphanumeric (A*n*V) in the Master File's USAGE and ACTUAL attributes.

`OFF`

Maps all fields defined as string in the XML schema to fixed-length alphanumeric (A) in the Master File's USAGE and ACTUAL attributes. This is the default.

Syntax: How to Change String Length

By default, when the Adapter for XML creates a synonym, it maps all fields defined as string in the XML schema to 10-byte alphanumeric in the Master File. You can change this default length using the SET FIELDLENGTH command

```
ENGINE XML XML SET FIELDLENGTH length
```

where:

`XML`

Indicates the adapter. You can omit this value if you previously issued the SET SQLENGINE command.

`length`

Is the length, in bytes, assigned to the ACTUAL and USAGE attributes of all fields defined in the XML schema as string. The maximum length is 3000.

Changing the Precision and Scale of Numeric Columns

You can alter the length and scale of numeric columns returned by a SELECT request to the server by creating different specifications in your login profile or in a stored procedure. The conversion settings are reflected in the Master File in the USAGE and ACTUAL formats of the fields generated by CREATE SYNONYM. This affects how the fields are processed and formatted by the server.

Tip: You can change this setting manually or from the Web Console.

For more information, see *How to Override the Default Precision and Scale* on page 96.

Conversion

The data in an XML document may reflect dates or numeric values, however, all the fields in a Master File synonym are set to the ALPHA data type.

Numeric Values

In order to enable arithmetic operations on numeric fields, the data type specified in the USAGE attribute of a numeric field needs to be modified, depending on the data in the XML document, to one of the following data types: Integer (I), Double Float (D) or Decimal (P). If the data type is modified to Double Float or Decimal, use scale and precision as necessary to describe the data in the XML document .

Furthermore, it is recommended that the length of the ALPHA data type specified in the ACTUAL attribute of the numeric field be modified to reflect the maximum length of the data in the XML document.

Dates in XML

In order to enable arithmetic operations on dates, the data type specified in the USAGE attribute of a date field needs to be modified, depending on the date format used in the XML document, to one of the following data types: YYMD, MDYY or DMYY.

Furthermore, the length of the ALPHA data type specified in the ACTUAL attribute of the date field needs to be modified to 10.

Note:

- Once you have set the data type in the Master File to one of the date data types, you must make sure that in each of the XML documents from which you report the date format is the same as the one you defined in the Master File. If the data types differ, an error will result.
- The date format in the answer set is not derived from the date format used in the XML document and is always set to YYYY/MM/DD.

Example: **Using Dates in XML**

If in the XML document you have the following format:	Then use USAGE=
1996-01-30	YYMD
01-30-1996	MDYY
30-01-1996	DMYY

Associating a Master File With an XML Document

There are two ways to associate a Master File with an XML document. The Master File can specify a DATASET attribute that points to the XML document or you can explicitly issue a FILEDEF command to associate a Master File with an XML document.

In order to properly code a FILEDEF command, you need to determine how the XML documents are organized. If the documents are in one file and are described by the same DTD, the organization is called a COLUMN. If there are various documents in one directory and all are described by the same DTD, then the organization is called a COLLECTION.

Source	Collection Organization	Column Organization
disk	supported	supported
http	not supported	supported
https	not supported	supported

Syntax: How to Associate a Master File With an XML Document Using FILEDEF

```
FILEDEF ddname source filename
```

where:

ddname

Is the logical reference name matching the desired Master File.

source

Is the location of the XML document. Possible values are http, https, and disk.

filename

For an XML Column, is the full path to the source file.

For an XML Collection, is the full path to the source directory. The directory name must be followed by the wildcard characters '*.*'.

The file name must start with 'http://' or 'https://' if the XML documents are to be read using an HTTP or HTTPS session.

Example: Issuing a FILEDEF Command for an XML Column

Note: If you are working on the Web Console or Data Management Console Create Synonym panes, see *Synonym Creation Parameters for XML* on page 2660 for details about how FILEDEFs are created.

For Windows, when you have the ordercol.xml document under C:\, use the following FILEDEF in your profile:

```
FILEDEF ordercol disk C:\ordercol.xml
```

For UNIX, when you have the ordercol.xml document under /usera/xml/, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK /usera/xml/ordercol.xml
```

For z/OS, when you have the ordercol.xml document in TEST.XML.DOC, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.XML.DOC'
```

For HTTP and HTTPS, when you have the ordercol.xml document under http://www.test.com/xmltest/ (or https://www.test.com/xmltest/), use the following FILEDEF in your profile:

```
FILEDEF order http http://www.test.com/xmltest/ordercol.xml
```

or

```
FILEDEF order http https://www.test.com/xmltest/ordercol.xml
```

Example: Issuing a FILEDEF Command for an XML Collection

Note: If you are working on the Web Console or Data Management Console Create Synonym panes, see *Synonym Creation Parameters for XML* on page 2660 for details about how FILEDEFs are created.

For Windows, when you have the order1.xml and order2.xml documents under C:\ORDERS, use the following FILEDEF in your profile:

```
FILEDEF order disk C:\orders\*.*
```

For UNIX, when you have the ordercol.xml document under /usera/orders, use the following FILEDEF in your profile:

```
FILEDEF ordercol DISK //'TEST.XML.DOC(dtd)'
```

For z/OS, when you have the order document in TEST.XML.DOC (DTD), use the following FILEDEF in your profile:

```
FILEDEF order DISK /usera/orders/*.*
```

Appendix A

XA Support

This topic describes XA Transaction Management and implementation specifics.

In this appendix:

XA Transaction Management

An application usually defines steps performed against any given resource in terms of *transactions*.

A *transaction* is a complete unit of work in which:

- Results are either all committed or all rolled back.
- The shared resource is transformed from one valid state to another.
- Changes to shared resources do not become visible outside the transaction until the transaction commits.
- The changes of commit are able to survive system or media failure.

The server has the ability to access multiple data sources and to carry out heterogeneous joins across different data sources. The transaction that describes work done by more than one Resource Manager is called a Global or Distributed Transaction.

Most relational and some non-relational resources are maintained by their Resource Managers. The X/Open Distributed Transaction Processing (DTP) model is a software architecture that allows multiple application programs to access resources provided by multiple resource managers, and allows their work to be coordinated into global transactions.

Read/write applications accessing CICS and IMS transactions, relational data sources, as well as CICS-controlled VSAM data sets are able to perform transactions managed in XA-compliant mode. The Transaction Coordinator component uses a two-phase commit protocol for completion of customer transactions across different database management systems.

Using the XA Transaction Management feature does not require any changes in existing ODBC, JDBC, or WebFOCUS Maintain applications. To activate the XA Transaction Management feature, the server has to be configured in Transaction Coordination Mode, using the Web Console configuration functions. In Transaction Coordination Mode, all requests performed against the listed XA-compliant DBMSs will be processed as XA-distributed transactions, and completed in two-phase commit mode. This guarantees the integrity of data modification on all of the involved DBMSs. It protects part of the data modifications from being committed on one DBMS and terminated on another.

Supported Interfaces

XA-compliant adapters are:

- Db2 (using CLI interface) on UNIX/Windows/Linux
- Db2 (using CLI Interface with RRS) on z/OS
- Oracle
- Microsoft SQL Server (using OLE DB interface)
- Informix
- Sybase ASE
- Ingres II

Non-XA-compliant adapters are:

- Adabas Stored Procedures
- CICS Transactions
- Informix CISAM
- Micro Focus CISAM
- IMS Transactions
- IMS
- VSAM CICS

Implementation

In order to adhere to the DTP model, the server-implemented Transaction Coordinator provides the ability to manage global transactions. Although the Transaction Coordinator is a private implementation, it fully adheres to the XA protocol in the boundaries of its implementation.

- Because the Transaction Coordinator is a private implementation, it does not have a public interface. There is no mechanism for foreign applications to use the Transaction Coordinator.
- Not every Resource Manager, even XA-compliant, is supported unless explicitly stated.

The implementation specifics of the Transaction Coordinator are:

- It is invoked by a keyword in edaserve.cfg (transaction_coordination_mode=on).
- Every unit of work initiated inside any agent will be started as a global transaction, within this agent.
- Coordination will apply to all currently supported XA-compliant interfaces.
- It cannot be turned off.
- All XA-compliant relational resources will participate, enabled implicitly.
- All non-relational resources can be explicitly disabled from participation.
- All transactions will be in the scope of a global transaction. Each commit/rollback will subsequently begin a transaction.
- No CREATE/DROP TABLE commands in XA mode.

Vendor Specifics

The implementation specifics for Db2 with CLI on UNIX, Windows, and z/OS are:

- The DB2CLI.INI file must have an entry in the common section

```
[COMMON]
DISABLEMULTITHREAD=1
```

- Only the global file, DB2CLI.INI, can be used. It must reside in:
  - For UNIX/NT, installation home.
  - For z/OS, exported by DSNAOINI variable.

The implementation specifics for Db2 with CLI on z/OS are:

- Requires RRS
- RRSAF in DB2CLI.INI

The implementation specific of MS SQL Server is:

- Required DTS

Note: If the client and the serve are on different machines, DTS must be installed on both machines.

The implementation specifics of Sybase are:

- The Sybase XA configuration file is needed.
- No bulk load/fast load functionality in XA, no messages are produced, and insert will operate in normal mode.
- Users can create their XA configuration file and point to it in the profile.

Syntax: How to Create the XA Configuration File

```
ENGINE SQLSYB SET XACONFIGFILE path
```

where:

path

Should include file name.

Appendix B

Aggregate Awareness Support

Aggregate Awareness substantially improves the efficiency of queries. In order to use this feature successfully, you need an understanding of the principles of relational adapter operation and knowledge of the RDBMS optimizer.

In this appendix:

Relational Adapters and Aggregated SQL Queries

Relational adapters construct optimized SQL queries based on a WebFOCUS request and the hierarchical structure of a file. A relational adapter always attempts to construct an SQL query that delegates most of the work (such as record screening, joins and aggregation) to the underlying relational engine. The ability of the interface to pass the join and all tests to the RDBMS is a prerequisite to constructing an aggregated SQL query.

If aggregation cannot be passed to the RDBMS, the trace file contains an explanation. Usually, the request or the file hierarchy can be corrected in such a way that aggregation can be passed to the RDBMS.

Aggregate Awareness in an RDBMS

When aggregation is delegated to the RDBMS, the RDBMS can perform additional optimization of the query by using a mechanism commonly called Aggregate Awareness. Aggregate Awareness is implemented in Db2, Teradata, Oracle, and some other RDBMSs.

The implementation is similar in most RDBMSs. Based on most common types of reports performed on a database, the database administrator creates one or more objects (named SUMMARY TABLES in Db2, JOIN INDEXES in Teradata, and MATERIALIZED VIEWS or SNAPSHOTS in Oracle) that are populated with pre-aggregated data. The size of the pre-aggregated object (summary table, join index, or snapshot) is usually much smaller than the combined size of the involved tables. The RDBMS optimizer evaluates the incoming queries and, if possible, reconstructs the incoming query so that the pre-aggregated data is used for forming the answer set. This significantly reduces both CPU time and I/O operations.

The aggregated SQL generated by the relational adapter is fully suitable for an RDBMS optimizer to use pre-aggregated data. However, Db2, Teradata, and Oracle optimizers have specific rules for when to use pre-aggregated data.

Syntax: How to Set Aggregate Awareness

```
SET AGGREGATE_AWARENESS {FRESH_ONLY|OLD_OK|OFF}
```

where:

`FRESH_ONLY`

Sets different values for the parameters associated with each RDBMS.

`OLD_OK`

Sets different values for the parameters associated with each RDBMS.

`OFF`

If no option is selected, the behavior of the target RDBMS is determined by the database configuration options. There is no default for this setting.

Reference: Usage Notes for Aggregate Awareness

Adapter-specific parameters for which values are set by the FRESHONLY and OLD_OK settings:

- For Db2, CURRENT REFRESH AGE and CURRENT QUERY OPTIMIZATION.
- For Oracle, QUERY_REWRITE_INTEGRITY and QUERY_REWRITE_ENABLED.

  These values can be set in the Server profile or as a part of an RPC using Direct Passthru.
- Teradata handles aggregate awareness internally.

RDBMS manuals should be reviewed for recommendations, but the most common checklist items are:

- Updating statistics on the pre-aggregated object and the master tables: (RUNSTATS in Db2, COLLECT STATISTICS in Teradata, ANALYZE TABLE in Oracle).
- Confirming compatibility of the generated SQL query and the query used for creating the pre-aggregated object. The most common items are compatibility of:
  - Aggregators (columns functions) and aggregation objects (column function arguments)
  - GROUP BY clauses
  - Joins

Appendix C

Cluster Join

This topic describes the Master File SEGSUF attribute, which enables you to create heterogeneous cluster joins, for example, between data sources of different types.

In this appendix:

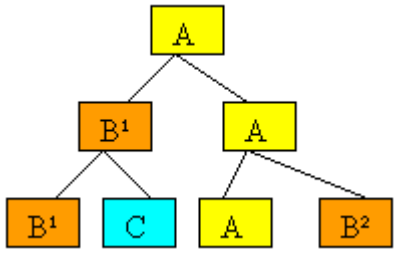
Embedded Joins

An embedded join is an equijoin that you define in a Master File. The name comes from the fact that the join definition is "embedded" in the Master File. (This is also known as a cluster join.)

You can define an embedded join between data sources:

- **Data sources of the same type.** For example, you can define an embedded join between two Db2 tables. This is known as a homogeneous embedded join.
- **Data sources of different types.** For example, you can define an embedded join between three Db2 tables, three CA-IDMS/DB records, and one IMS segment. This is known as a heterogeneous embedded join.

  In the following heterogeneous embedded join, A = a Db2 table, B = a CA-IDMS/DB record, and C = an IMS segment.



  In a Master File, each segment declaration describes one RDBMS table or view, or one record. You can include up to 64 segments in an embedded join.

  The entire join structure described by the Master File is referred to as a *tree*.

A *subtree* is a segment and any directly related descendant segments of the same data source type. A *root subtree* is the subtree at the top of the tree.

There are four subtrees here:

- A-A-A
- B^1-B^1
- B^2
- C

Note that B^1-B^1 and B^2 are separate subtrees because, even though they are of the same data source type, they are not directly related.

The root subtree is A-A-A.

Embedded Join Master Files

Each embedded join segment declaration describes one RDBMS table or view, or one non-RDBMS record. You can include up to 64 segments.

If several Master Files (used only with TABLE requests) include the same table, you can avoid repeating the same description multiple times. Describe the table in one of the Master Files, and use the CRFILE attribute in the other Master Files to access the existing description.

Syntax: How to Define an Equijoin in the Master File

An embedded equijoin uses the PARENT segment attribute to identify the relationship between tables and/or records.

```
FILENAME=mtname, SUFFIX=rootsuffix   [,$]
 SEGNAME=table1, SEGTYPE= {S0} [,CRFILE=crfile1] [,$]
  FIELD=name,...,$
  .
  .
  .
 SEGNAME=table2, [SEGSUF=branchsuffix,] SEGTYPE=relationship,
  PARENT=table1 [,CRFILE=crfile2][,$]
   FIELD=name,...,$
  .
  .
  .
```

where:

mtname

Is the one-character to eight-character name of the embedded join Master File.

rootsuffix

When the Master File describes:

- A homogeneous embedded join, *rootsuffix* specifies which adapter to use to access the data source.
- A heterogeneous embedded join, *rootsuffix* specifies which adapter to use to access the highest-level subtree of the join. That is, it specifies which adapter to use to access the tables or records represented by the root segment and all directly-related descendant segments that are of the same DBMS type.

table1

Is the SEGNAME value for the root table or record in the embedded join. If this segment references a remote segment description, table1 must be identical to the SEGNAME from the Master File that contains the full definition of the remote data source.

name

Is any field name.

table2

Is the SEGNAME value for the related table or record. If this segment references a remote segment description, *table2* must be identical to the SEGNAME from the Master File that contains the full definition of the remote data source.

SEGSUF

Is used in a heterogeneous embedded join to identify the beginning of a new subtree (other than the root subtree), and to specify which adapter to use to access that subtree. A subtree is a segment (representing a table or record) and all directly related descendant segments of the same data source type. You specify SEGSUF in the highest-level segment of the subtree.

branchsuffix

Specifies the value of SEGSUF.

relationship

Indicates the type of relationship between the table or record and its parent. Possible values are:

S0 indicates that the related table is in a one-to-many or many-to-many (non-unique) relationship with the table or record named as its parent.

U indicates that the related table or record is in a one-to-one or a many-to-one (unique) relationship with the table or record named as its parent.

KL references a remote segment description. Indicates that the related table or record is in a one-to-many or many-to-many (non-unique) relationship with the table or record named as its parent.

KLU references a remote segment description. Indicates that the related table or record is in a one-to-one or a many-to-one (unique) relationship with the table or record named as its parent.

crfile1

References a remote segment description. Indicates the name of the Master File that contains the full definition of table1.

crfile2

References a remote segment description. Indicates the name of the Master File that contains the full definition of table2.

Appendix D

Translating COBOL File Descriptions

This topic describes how synonyms are created from COBOL File Descriptions by the following adapters:

- C-ISAM
- Flat and Delimited Flat Files
- CICS Transactions
- IMS
- IMS Transactions
- Millennium
- RMS
- VSAM and VSAM CICS

The information in this appendix is intended to supplement synonym-creation information in the individual adapter chapters.

In this appendix:

Creating Synonyms From COBOL File Descriptions

Synonyms define unique names (or aliases) for each file that is accessible from a server. Synonyms are useful because they hide location information and the identity of the underlying data source from client applications. They also provide support for extended metadata features of the server such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server.

The CREATE SYNONYM command uses COBOL FD translation capabilities to translate the COBOL file definition into an internal metadata format from which a Master File is created.

Controlling the Translation of a COBOL File Description

You can take advantage of a variety of options to control the translation of a COBOL File Description to a Master File. These options are available from the Create Synonym panes in the adapters that use COBOL FDs. If customization options are not selected. default translation setting are applied.

This topic provides supplementary information for several customization options.

Customization Options for COBOL File Descriptions

You can customize how a COBOL FD is translated by selecting *Customize options* in the Synonym creation pane. The following options are added to the right pane:

Parameter	Definition
On Error	Choose: ❑ *Continue* to continue generating the Master File when an error occurs. Continue is the default value. ❑ *Abort* to stop generating the Master File when an error occurs. ❑ *Comment* to produce a commented Master File when an error occurs.
Hyphens as	Choose: ❑ *No* to remove all hyphens in the COBOL name from the Master File field names. ❑ *Yes* to replace all hyphens in the COBOL name with the underscore character. *Yes* is the default value.

Parameter	Definition
Redefines	You may treat COBOL REDEFINE fields in one of three ways. Choose: ❑ *Segments* to describe REDEFINE fields as segments in the Master File. Segments is the default value. ❑ *Comments* to describe REDEFINE fields as comments in the Master File. ❑ *None* to exclude REDEFINE fields altogether.
Occurs as	Choose *Segments* to describe OCCURS structures as segments. Otherwise, choose *Field*. Segments is the default value.
Alignment	Choose: ❑ *Yes* to insert slack bytes into a record to ensure alignment of numeric fields. ❑ *No* to generate Master Files without alignment of slack bytes. No is the default value.
Number of Hyphens to skip	FD Translator removes characters from the left, up to and including the Nth hyphen (depending on the value of N chosen in the menu). ❑ *0* retains the entire COBOL name. *0* is the default value. ❑ *All* removes all prefixes.
Order fields	Choose: ❑ *Yes* to generate Order fields in a Master File. ❑ *No* to generate a Master File without Order fields. *No* is the default value.
Level 88 as	Choose: ❑ *Comment* to include COBOL Level 88 fields as comments in the Master Files. ❑ *Skip* to exclude level 88 fields. *Skip* is the default value.

Parameter	Definition
Zoned Numeric Fields	Sets how zoned numeric values will be stored.
Numeric Field Edit Options	
Zeroes	Choose: ❑ *Suppress* to suppress printing of the digit zero for a field whose value is zero. ❑ *Display* to display leading zeroes, for example, 00124. ❑ *None* for no formatting.
Negative value	Choose: ❑ *Bracket* to bracket negative values, for example, (1234). ❑ *Credit* to credit negative values, for example, 1234 CR. ❑ *None* for no formatting.
Dollar Sign	Choose: ❑ *Floating* to display a floating dollar sign and commas, for example, $1,123. ❑ *Fixed* to display a fixed dollar sign and commas, for example, $ 1,123. ❑ *None* for no formatting.
Separate Thousands	Choose: ❑ *Comma* to include commas where appropriate. ❑ *None* for no formatting.

REDEFINE Fields

You may treat COBOL REDEFINE fields in one of three ways:

- Describe REDEFINEs as segments in the Master File (enter S for segments).
- Describe REDEFINEs as comments in the Master File (enter C for comments).
- Exclude REDEFINEs altogether (enter N for none).

Describing REDEFINEs as segments gives you immediate access to all first-level REDEFINEs but may still require user customization. Note that nested REDEFINEs are described as comments, even with this option. The Segment option is only available in software releases that support it. Including REDEFINEs as comments allows you to customize the Master File output to select the redefinitions of your choice.

Fields are described as comments with a dollar sign ($) in column one. If you wish to use the redefinition of a field instead of the original definition, delete the first definition, or insert dollar signs ($) in column one, and remove the dollar signs ($) from the redefinition.

Example: Generating REDEFINE Fields

Consider the following COBOL input description:

```
01  CLIENT-REC.
    02  NAME-ADDR-AREA  PIC X(57).
    02  NAME-AND-ADDR
        REDEFINES NAME-ADDR-AREA.
         03  NAME          PIC X(20).
         03  STREET        PIC X(15).
         03  CITY          PIC X(15).
         03  STATE         PIC X(2).
         03  ZIP           PIC X(5).
         03  ZIP_NUMERIC   REDEFINES ZIP PIC 9(5).
```

The following three Master Files are generated from this input, depending on the value entered for Generate REDEFINE Fields.

For a value of S:

```
FILE=REDEFS,                      SUFFIX=FIX,                    $
SEGNAME=CLIENSEG,     SEGTYPE=S0,                                $
   GROUP=CLIENTREC,    ALIAS=E01,   USAGE=A57,      ACTUAL=A57,  $
   FIELD=NAMEADDRAREA, ALIAS=E02,   USAGE=A57,      ACTUAL=A57,  $
SEGNAME=NAMEASEG,     SEGTYPE=U,   PARENT=CLIENSEG,
OCCURS=1,             POSITION=NAMEADDRAREA,                     $
   GROUP=NAMEANDADDR,  ALIAS=E03,   USAGE=A57,      ACTUAL=A57,  $
   FIELD=NAME,         ALIAS=E04,   USAGE=A20,      ACTUAL=A20,  $
   FIELD=STREET,       ALIAS=E05,   USAGE=A15,      ACTUAL=A15,  $
   FIELD=CITY,         ALIAS=E06,   USAGE=A15,      ACTUAL=A15,  $
   FIELD=STATE,        ALIAS=E07,   USAGE=A2,       ACTUAL=A2,   $
   FIELD=ZIP,          ALIAS=E08,   USAGE=A5,       ACTUAL=A5,   $
$  FIELD=ZIP_NUMERIC,  ALIAS=E09,   USAGE=P5,       ACTUAL=Z5,   $
```

For a value of C:

```
FILE=REDEFC,                      SUFFIX=FIX,                    $
SEGNAME=CLIENSEG,     SEGTYPE=S0,                                $
   GROUP=CLIENTREC,    ALIAS=E01,   USAGE=A57,      ACTUAL=A57,  $
   FIELD=NAMEADDRAREA, ALIAS=E02,   USAGE=A57,      ACTUAL=A57,  $
$ GROUP=NAMEANDADDR,   ALIAS=E03,   USAGE=A57,      ACTUAL=A57,  $
$   FIELD=NAME,        ALIAS=E04,   USAGE=A20,      ACTUAL=A20,  $
$   FIELD=STREET,      ALIAS=E05,   USAGE=A15,      ACTUAL=A15,  $
$   FIELD=CITY,        ALIAS=E06,   USAGE=A15,      ACTUAL=A15,  $
$   FIELD=STATE,       ALIAS=E07,   USAGE=A2,       ACTUAL=A2,   $
$   FIELD=ZIP,         ALIAS=E08,   USAGE=A5,       ACTUAL=A5,   $
$   FIELD=ZIP_NUMERIC, ALIAS=E09,   USAGE=P5,       ACTUAL=Z5,   $
```

For a value of N:

```
FILE=REDEFN,                      SUFFIX=FIX,                    $
SEGNAME=CLIENSEG,     SEGTYPE=S0,                                $
   GROUP=CLIENTREC,    ALIAS=E01,   USAGE=A57,      ACTUAL=A57,  $
   FIELD=NAMEADDRAREA, ALIAS=E02,   USAGE=A57,      ACTUAL=A57,  $
```

LEVEL 88 as Comments

Level 88 fields in the COBOL FD associate particular values with a field name. Level 88 fields and values are not required in the Master File. However, you can include them as comments. Enter Y (for yes) to include them or N (for no) to exclude them. Including Level 88 fields lets you easily identify values for RECTYPEs for those files that require Level 88 fields. (You must manually add the RECTYPE and value. Because many Level 88 fields may occur in a COBOL FD, it is impossible to successfully determine the appropriate RECTYPE fields and values.) Since Level 88 field names typically describe the values they represent, including them as comments is also a way to document the Master File.

Example: Generating LEVEL 88 as Comments

Consider the following COBOL input description:

```
01  EMPL-REC.
    02  EMPL-SOC-SEC-NUM          PIC S9(8) COMP-3.
    02  EMPL-STATUS               PIC X(1).
        88  EMPL-ACTIVE                    VALUE 'A'.
        88  EMPL-INACTIVE                  VALUE 'I'.
        88  EMPL-RETIRED                   VALUE 'R'.
```

The following two Master Files are generated from this input, depending on the value entered for Generate LEVEL 88 as Comments.

For a value of Y:

```
FILE=LEV88Y,                        SUFFIX=FIX,                    $
SEGNAME=RECSEG,       SEGTYPE=S0,                                  $
   FIELD=SOCSECNUM,    ALIAS=E01,   USAGE=P9,       ACTUAL=P5,     $
   FIELD=STATUS,       ALIAS=E02,   USAGE=A1,       ACTUAL=A1,     $
$       ACTIVE, VALUE 'A'.                                         $
$       INACTIVE, VALUE 'I'.                                       $
$       RETIRED, VALUE 'R'.                                        $
```

The Level 88 fields are indented under the field they describe (STATUS). The values are left-justified to include the maximum information in the Master File, when many values are in the COBOL record. If the COBOL Level 88 values exceed one line in the FD, only the first line will be included in the Master File.

For a value of N:

```
FILE=LEV88N,                        SUFFIX=FIX,                    $
SEGNAME=RECSEG,       SEGTYPE=S0,                                  $
   FIELD=SOCSECNUM,    ALIAS=E01,   USAGE=P9,       ACTUAL=P5,     $
   FIELD=STATUS,       ALIAS=E02,   USAGE=A1,       ACTUAL=A1,     $
```

OCCURS as Segments

COBOL FD translation supports all forms of COBOL OCCURS structures: fixed, variable, and nested. You may describe OCCURS structures as segments in the Master File or as individual, numbered fields. Enter Y (for yes) to describe OCCURS structures as segments or N (for no) otherwise.

Fixed and nested repeating groups described as segments use the OCCURS clause to describe the number of occurrences of the group. For fixed repeating groups, the OCCURS value is the number of occurrences. For variable OCCURS, the value is the name of the COBOL DEPENDING ON field. The POSITION attribute is used to describe their location in the record, which is reserved in the Master File with an internally generated POSITION field. All internal POSITION fields generated by the translation end with POSN.

Fixed repeating groups described as numbered fields have a unique number appended to the COBOL field name, once for each occurrence. Because the number of occurrences of a variably occurring group is unknown, they cannot be described individually. In fact, if any one of the repeating groups is variable, they are all described as segments, including fixed occurs, regardless of your menu selection.

Example: Simple Fixed OCCURS

Consider the following COBOL input description:

```
01  TBL-REC.
    02  TBL-ENTRY OCCURS 2 TIMES.
        03  TBL-AMT-A                  PIC S9(5).
        03  TBL-AMT-B                  PIC S9(5).
```

The translation can generate the following two Master Files from this input, depending on the value entered for Describe OCCURS as Segments.

For a value of Y:

```
FILE=OCCURSFY,                    SUFFIX=FIX,                    $
SEGNAME=TBLRESEG,      SEGTYPE=S0,                               $
   GROUP=TBLREC,        ALIAS=E01,   USAGE=A20,      ACTUAL=A20,  $
   FIELD=TBLENSEGPOSN, ALIAS=E02,   USAGE=A20,      ACTUAL=A20,  $
SEGNAME=TBLENSEG,      SEGTYPE=S0,  PARENT=TBLRESEG,
OCCURS=2,              POSITION=TBLENSEGPOSN,                    $
   GROUP=TBLENTRY,      ALIAS=E03,   USAGE=A16,      ACTUAL=A10,  $
   FIELD=TBLAMTA,       ALIAS=E04,   USAGE=P6,       ACTUAL=Z5,   $
   FIELD=TBLAMTB,       ALIAS=E05,   USAGE=P6,       ACTUAL=Z5,   $
```

For a value of N:

```
FILE=OCCURSFN,                    SUFFIX=FIX,                    $
SEGNAME=TBLRESEG,      SEGTYPE=S0,                               $
   GROUP=TBLREC,        ALIAS=E01,   USAGE=A32,      ACTUAL=A20,  $
   GROUP=TBLENTRY1,     ALIAS=E02,   USAGE=A16,      ACTUAL=A10,  $
   FIELD=TBLAMTA1,      ALIAS=E03,   USAGE=P6,       ACTUAL=Z5,   $
   FIELD=TBLAMTB1,      ALIAS=E04,   USAGE=P6,       ACTUAL=Z5,   $
   GROUP=TBLENTRY2,     ALIAS=E05,   USAGE=A16,      ACTUAL=A10,  $
   FIELD=TBLAMTA2,      ALIAS=E06,   USAGE=P6,       ACTUAL=Z5,   $
   FIELD=TBLAMTB2,      ALIAS=E07,   USAGE=P6,       ACTUAL=Z5,   $
```

Example: Variable OCCURS

Consider the following COBOL input description:

```
01  TABLE-REC.
    02  TABLE-COUNT            PIC S9(2) COMP.
    02  TABLE-ENTRY
        OCCURS 1 TO 10 TIMES
        DEPENDING ON TABLE-COUNT.
        03  TABLE-AMT-A        PIC S9(5).
        03  TABLE-AMT-B        PIC S9(5).
```

The translation generates the following Master Files from this input when either Y or N is entered for Describe OCCURS as Segments:

```
FILE=OCCURSVY,                    SUFFIX=FIX,                    $
SEGNAME=TABLESEG,     SEGTYPE=S0,                                $
   FIELD=TABLECOUNT,   ALIAS=E01,   USAGE=I3,       ACTUAL=I2,    $
SEGNAME=TABLESE2,     SEGTYPE=S0,  PARENT=TABLESEG,
OCCURS=TABLECOUNT,                                               $
   GROUP=TABLEENTRY,   ALIAS=E02,   USAGE=A16,      ACTUAL=A10,   $
   FIELD=TABLEAMTA,    ALIAS=E03,   USAGE=P6,       ACTUAL=Z5,    $
   FIELD=TABLEAMTB,    ALIAS=E04,   USAGE=P6,       ACTUAL=Z5,    $
```

The fields to be reported from (TABLEAMTA and TABLEAMTB) are within the OCCURS segment. The number of table occurrences is determined automatically by the TABLECOUNT field.

Example: Nested OCCURS

Consider the following COBOL input description:

```
01  TABLE-REC.
    02  TABLE-LEVEL-A OCCURS 2.
        03  TABLE-LEVEL-B OCCURS 3.
            04  TABLE-AMT              PIC S9(5) COMP-3.
```

The translation can generate the following two Master Files from this input, depending on the value entered for Describe OCCURS as Segments:

For a value of Y:

```
FILE=OCCURSNY,                    SUFFIX=FIX,                    $
SEGNAME=TABLESEG,     SEGTYPE=S0,                                $
   GROUP=TABLEREC,     ALIAS=E01,   USAGE=A18,      ACTUAL=A18,   $
   FIELD=TABLESE2POSN, ALIAS=E02,   USAGE=A18,      ACTUAL=A18,   $
SEGNAME=TABLESE2,     SEGTYPE=S0,  PARENT=TABLESEG,
OCCURS=2,             POSITION=TABLESE2POSN,                     $
   GROUP=TABLELEVELA,  ALIAS=E03,   USAGE=A9,       ACTUAL=A9,    $
   FIELD=TABLESE3POSN, ALIAS=E04,   USAGE=A9,       ACTUAL=A9,    $
SEGNAME=TABLESE3,     SEGTYPE=S0,  PARENT=TABLESE2,
OCCURS=3,             POSITION=TABLESE3POSN,                     $
   GROUP=TABLELEVELB,  ALIAS=E05,   USAGE=A8,       ACTUAL=A3,    $
   FIELD=TABLEAMT,     ALIAS=E06,   USAGE=P6,       ACTUAL=P3,    $
```

The translation defines the nested table structure with nested segments. You need only refer to the final reporting field, TABLEAMT.

For a value of N:

```
FILE=OCCURSNN,                      SUFFIX=FIX,                        $
SEGNAME=TABLESEG,      SEGTYPE=S0,                                     $
   GROUP=TABLEREC,      ALIAS=E01,   USAGE=A48,       ACTUAL=A18,     $
   GROUP=TABLELEVELA1, ALIAS=E02,   USAGE=A24,       ACTUAL=A9,      $
   GROUP=TABLELEVELB1, ALIAS=E03,   USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT11,   ALIAS=E04,   USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB2, ALIAS=E05,   USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT12,   ALIAS=E06,   USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB3, ALIAS=E07,   USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT13,   ALIAS=E08,   USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELA2, ALIAS=E09,   USAGE=A24,       ACTUAL=A9,      $
   GROUP=TABLELEVELB4, ALIAS=E10,   USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT21,   ALIAS=E11,   USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB5, ALIAS=E12,   USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT22,   ALIAS=E13,   USAGE=P6,        ACTUAL=P3,      $
   GROUP=TABLELEVELB6, ALIAS=E14,   USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT23,   ALIAS=E15,   USAGE=P6,        ACTUAL=P3,      $
```

ORDER Field

If you wish to select specific occurrences for reporting, add an ORDER field to the Master File as the last entry of the OCCURS segment:

```
FIELD=fieldname,    ALIAS=ORDER, USAGE=I4,       ACTUAL=I4, $
```

The value for ALIAS must be ORDER, the value for ACTUAL must be I4, the field name is arbitrary, and the value for USAGE must be an integer (I) but can be of any length from 1 to 9. The ORDER field is automatically populated.

Example: ORDER Field Sequence With Nested OCCURS

The following example adds the ORDER field SEQUENCE to the nested fixed OCCURS segments 2 and 3:

```
FILE=OCCURSNY,                    SUFFIX=FIX,                          $
SEGNAME=TABLESEG,      SEGTYPE=S0,                                     $
   GROUP=TABLEREC,      ALIAS=E01,    USAGE=A18,       ACTUAL=A18,     $
   FIELD=TABLESE2POSN, ALIAS=E02,    USAGE=A18,       ACTUAL=A18,     $
SEGNAME=TABLESE2,      SEGTYPE=S0,  PARENT=TABLESEG,
OCCURS=2,              POSITION=TABLESE2POSN,                          $
   GROUP=TABLELEVELA,  ALIAS=E03,    USAGE=A9,        ACTUAL=A9,      $
   FIELD=TABLESE3POSN, ALIAS=E04,    USAGE=A9,        ACTUAL=A9,      $
   FIELD=SEQUENCEA,    ALIAS=ORDER, USAGE=I4,        ACTUAL=I4,      $
SEGNAME=TABLESE3,      SEGTYPE=S0,  PARENT=TABLESE2,
OCCURS=3,              POSITION=TABLESE3POSN,                          $
   GROUP=TABLELEVELB,  ALIAS=E05,    USAGE=A8,        ACTUAL=A3,      $
   FIELD=TABLEAMT,     ALIAS=E06,    USAGE=P6,        ACTUAL=P3,      $
   FIELD=SEQUENCEB,    ALIAS=ORDER, USAGE=I4,        ACTUAL=I4,      $
```

Zoned Numeric Field Usage

A COBOL FD may describe alphanumeric characters as zoned decimal numeric data using the PICTURE 9(n) clause. This clause is commonly used for numeric string data on which no arithmetic operations is performed, such as phone numbers, identification numbers, and dates. You can describe these fields in packed numeric or alphanumeric USAGE format (enter the values P or A, respectively). The ACTUAL format is described as zoned (Z).

If you intend to use COBOL numeric display fields for summing or in mathematical operations, use the packed option. If you intend to use these elements for display only, use the alphanumeric option. Only simple numeric display elements of the format PICTURE 9(n) are subject to this feature. Signed elements (PICTURE S9(n)) and elements with decimal positions (PICTURE 9(n)V(m)) are always described as packed.

Example: Zoned Numeric Field Usage

Consider the following COBOL input description:

```
01 EMPL-REC.
      02 EMPL-SOC-SEC-NUM         PIC 9(8).
      02 EMPL-HIRE-DATE.
         03 EMPL-HIRE-YEAR        PIC 9(2).
         03 EMPL-HIRE-MONTH       PIC 9(2).
         03 EMPL-HIRE-DAY         PIC 9(2).
      02 EMPL-SICK-DAYS-ALLOWED   PIC S9(2).
      02 EMPL-SICK-DAYS-TAKEN     PIC 9(2).
      02 EMPL-YTD-HOURS-WORKED    PIC 9(4)V9(2).
```

The translation can generate the following two Master Files from this input, depending on the value entered for Zoned Numeric Field Usage.

For a value of P:

```
FILE=ZONEDP,                      SUFFIX=FIX,                    $
SEGNAME=RECSEG,        SEGTYPE=S0,                               $
   GROUP=REC,           ALIAS=E01,   USAGE=A56,      ACTUAL=A24,  $
   FIELD=SOCSECNUM,     ALIAS=E02,   USAGE=P8,       ACTUAL=Z8,   $
   GROUP=HIREDATE,      ALIAS=E03,   USAGE=A24,      ACTUAL=A6,   $
   FIELD=HIREYEAR,      ALIAS=E04,   USAGE=P2,       ACTUAL=Z2,   $
   FIELD=HIREMONTH,     ALIAS=E05,   USAGE=P2,       ACTUAL=Z2,   $
   FIELD=HIREDAY,       ALIAS=E06,   USAGE=P2,       ACTUAL=Z2,   $
   FIELD=SICKDAYSALLO,  ALIAS=E07,   USAGE=P3,       ACTUAL=Z2,   $
   FIELD=SICKDAYSTAKE,  ALIAS=E08,   USAGE=P2,       ACTUAL=Z2,   $
   FIELD=YTDHOURSWORK,  ALIAS=E09,   USAGE=P7.2,     ACTUAL=Z6,   $
```

For a value of A:

```
FILE=ZONEDA,                      SUFFIX=FIX,                    $
SEGNAME=RECSEG,        SEGTYPE=S0,                               $
   GROUP=REC,           ALIAS=E01,   USAGE=A32,      ACTUAL=A24,  $
   FIELD=SOCSECNUM,     ALIAS=E02,   USAGE=A8,       ACTUAL=Z8,   $
   GROUP=HIREDATE,      ALIAS=E03,   USAGE=A6,       ACTUAL=A6,   $
   FIELD=HIREYEAR,      ALIAS=E04,   USAGE=A2,       ACTUAL=Z2,   $
   FIELD=HIREMONTH,     ALIAS=E05,   USAGE=A2,       ACTUAL=Z2,   $
   FIELD=HIREDAY,       ALIAS=E06,   USAGE=A2,       ACTUAL=Z2,   $
   FIELD=SICKDAYSALLO,  ALIAS=E07,   USAGE=P3,       ACTUAL=Z2,   $
   FIELD=SICKDAYSTAKE,  ALIAS=E08,   USAGE=A2,       ACTUAL=Z2,   $
   FIELD=YTDHOURSWORK,  ALIAS=E09,   USAGE=P7.2,     ACTUAL=Z6,   $
```

The two generated Master Files contains zoned decimal fields that are truly used in numeric and alphanumeric formats, but the translation cannot distinguish between the two cases. You must select the option most appropriate to your situation. You may edit the resulting Master File to change the formats to packed or alphanumeric as necessary. The USAGE length of any GROUP fields that contain these zoned fields must also be changed, according to the format conversion rules in *General Format Conversion Notes* on page 2712.

Numeric Field Edit Options

The COBOL FD translation automatically adds any edit options that you supply to all numeric fields in the generated Master File. Edit options affect how the numeric data is displayed. The options are as follows:

S - Suppresses printing of the digit zero for a field whose value is zero.

C - Includes commas where appropriate, for example, 1,234.

B - Brackets negative values, for example, (1234).

R - Credits negative values, for example, 1234 CR.

M - Displays a floating dollar sign with commas, for example, $1,234.

N - Displays a fixed dollar sign with commas, for example, $ 1,234.

L - Displays leading zeroes, for example, 001234.

You may enter up to five edit options. You may leave this option blank if you do not wish to edit numeric fields. Edit options may be entered in any order and combination except for the pairs SL, MN, and BR, which are mutually exclusive.

Example: **Numeric Field Edit Options**

Consider the following COBOL input description:

```
01  PAYROLL-REC.
      05  PAYROLL-NAME.
          10  PAYROLL-LAST-NAME   PIC X(20).
          10  PAYROLL-FIRST-NAME  PIC X(10).
      05  PAYROLL-AMT-FIELDS.
          10 PAYROLL-AMT-CURR-YR  PIC S9(7)V99 COMP-3.
          10 PAYROLL-AMT-PREV-YR  PIC S9(7)V99 COMP-3.
```

The translation generates the following Master Files from this input when MB is entered for Numeric Field Edit Options:

```
FILE=EDITOPT,                    SUFFIX=FIX,                    $
SEGNAME=RECSEG,       SEGTYPE=S0,                               $
   GROUP=REC,          ALIAS=E01,   USAGE=A46,      ACTUAL=A40,  $
   GROUP=NAME,         ALIAS=E02,   USAGE=A30,      ACTUAL=A30,  $
   FIELD=LASTNAME,     ALIAS=E03,   USAGE=A20,      ACTUAL=A20,  $
   FIELD=FIRSTNAME,    ALIAS=E04,   USAGE=A10,      ACTUAL=A10,  $
   GROUP=AMTFIELDS,    ALIAS=E05,   USAGE=A16,      ACTUAL=A10,  $
   FIELD=AMTCURRYR,    ALIAS=E06,   USAGE=P11.2MB,  ACTUAL=P5,   $
   FIELD=AMTPREVYR,    ALIAS=E07,   USAGE=P11.2MB,  ACTUAL=P5,   $
```

Field Name Information

The translation generates field names in the Master File from COBOL field names using the following:

1. The Skip 'n' Hyphens option removes characters from the left, up to and including the nth hyphen (depending on the value of n you entered on the menu). It then either removes all remaining hyphens or replaces them with underscores, depending on the setting of the menu option Remove Hyphens or Use Underbars. Finally, if the Field Name Length option has been set to 12, only the 12 leftmost characters are used.
2. If the name duplicates a previously-generated field name, a qualifier is added to make it unique. For example, given the following COBOL structure

   ```
   02  EMPL-PORTION.
       03  SOC-SEC-NUM  PIC 9(9).
   02  MGR-PORTION.
       03  SOC-SEC-NUM  PIC 9(9).
   ```

   SOCSECNUM is generated from the first elementary field name and SOCSECNUM2 from the second elementary field name.

The following table illustrates the results of the possible combinations of menu options that control field name generation.

COBOL Field Name: PAYROLL-REC-IN-FICA			
12	Remove Hyphens	0	PAYROLLRECIN
		1	RECINFICA
		2	INFICA
		3	FICA
	Use Underbars	0	PAYROLL_REC
		1	REC_IN_FICA
		2	IN_FICA
		3	FICA
30	Remove Hyphens	0	PAYROLLRECINFICA
		1	RECINFICA
		2	INFICA
		3	FICA
	Use Underbars	0	PAYROLL_REC_IN_FICA
		1	REC_IN_FICA
		2	IN_FICA
		3	FICA

General Format Conversion Notes

Format conversion is the process of defining the ACTUAL and USAGE formats based on the COBOL format. The translation performs the conversion as described in the following table:

COBOL Format	ACTUAL	USAGE
PICTURE X(n)	An	An
PICTURE 9(n)	Zn (packed option)	Pn (packed option)
PICTURE 9(n)	An (alpha option)	An (alpha option)
PICTURE S9(n)	Zn	Pn+1
PICTURE 9(n)V9(m)	Zn+m	Pn+m+1.m
PICTURE S9(n)V9(m)	Zn+m	Pn+m+2.m
PICTURE 9(n) COMP (1 # n # 4)	I2	I9
PICTURE 9(n) COMP (5 # n # 9)	I4	I9
PICTURE 9(n) COMP (n > 9)	A8	A8
COMP-1	F4	F8
COMP-2	D8	D15
PICTURE 9(n) COMP-3	P(n+2/2)	Pn
PICTURE S9(n) COMP-3	P(n+2/2)	Pn+1
PICTURE 9(n)V9(m) COMP-3	P(n+m+2/2)	Pn+m+1.m
PICTURE S9(n)V9(m) COMP-3	P(n+m+2/2)	Pn+m+2.m

The format conversions are subject to the following limitations:

- Alphanumeric fields are limited to 256 characters. Elementary fields that exceed 256 characters are truncated to 256 characters and have filler fields inserted to provide for the total length. Group fields that exceed 256 characters are commented.
- Unsigned zoned fields that exclude a decimal point are described with a USAGE of packed (P) or alphanumeric (A), depending on the value entered for the Zoned Numeric Field Usage option.

- COMP-4 binary fields are described the same as COMP fields.
- Binary fields that exceed 9 digits in the picture clause are described with ACTUAL format A8 and USAGE format A8.
- The maximum length of the packed USAGE format depends on the software release that uses the generated Master File. For releases that support long packed fields (16 or more digits), the maximum ACTUAL length is 16 bytes and the maximum USAGE length is 31 digits (or 32 characters including a decimal point and sign). For earlier releases, the maximum ACTUAL length is 8 bytes and the maximum USAGE length is 15 total digits, including decimal and sign. The number of decimal places is limited to one fewer than the total USAGE length.
- In general, the USAGE length of packed fields is calculated as the sum of the digits to the left of the decimal (n), the number of decimal positions (m), one for the leading minus sign if present (S), and one for the decimal (V) if present.
- The ACTUAL and USAGE formats for GROUP fields are always alphanumeric (A). The ACTUAL length is the sum of the ACTUAL lengths of its components. The USAGE length is the sum of the following:
- The USAGE lengths of all the alphanumeric (A) components.
- 16 for each long packed (P) component.
- 8 for each short packed (P) and double precision (D) component.
- 4 for each integer (I) and floating point (F) component.

Multiple Records as Input

The translation generates a new segment for each COBOL record description (01-level field) it receives as input. When multiple records are present, an additional DUMMY segment is created that is used as the common parent for the record descriptions. Whenever you submit more than one 01-level record as input, you must manually add RECTYPE fields to identify the different record types.

Example: Multiple Records as Input

Consider the following COBOL input description:

```
01  HDR-REC.
       02  HDR-KEY            PIC X(1).
           88 HDR-VALUE       VALUE 'H'.
       02  HDR-DATA           PIC X(10).
01  DET-REC.
       02  DET-KEY            PIC X(1).
           88 DTL-VALUE       VALUE 'D'.
       02  DTL-DATA           PIC X(10).
```

From this input, the following Master File with multiple segments is created:

```
FILE=MULT01,                        SUFFIX=FIX,                          $
SEGNAME=DUMMY,         SEGTYPE=S0,                                       $
   FIELD=,              ALIAS=,      USAGE=A1,       ACTUAL=A1,          $
SEGNAME=HDRRESEG,      SEGTYPE=S0,  PARENT=DUMMY,                        $
   GROUP=HDRREC,        ALIAS=E01,   USAGE=A11,      ACTUAL=A11,         $
   FIELD=HDRKEY,        ALIAS=E02,   USAGE=A1,       ACTUAL=A1,          $
   $     HDRVALUE, VALUE 'H'.                                            $
   FIELD=HDRDATA,       ALIAS=E03,   USAGE=A10,      ACTUAL=A10,         $
SEGNAME=DETRESEG,      SEGTYPE=S0,  PARENT=DUMMY,                        $
   GROUP=DETREC,        ALIAS=E04,   USAGE=A11,      ACTUAL=A11,         $
   FIELD=DETKEY,        ALIAS=E05,   USAGE=A1,       ACTUAL=A1,          $
   $     DTLVALUE, VALUE 'D'.                                            $
   FIELD=DTLDATA,       ALIAS=E06,   USAGE=A10,      ACTUAL=A10,         $
```

After editing to add the RECTYPE information, the completed Master File is:

```
FILE=MULT01,                        SUFFIX=FIX,                          $
SEGNAME=DUMMY,         SEGTYPE=S0,                                       $
   FIELD=,              ALIAS=,      USAGE=A1,       ACTUAL=A1,          $
SEGNAME=HDRRESEG,      SEGTYPE=S0,  PARENT=DUMMY,                        $
   GROUP=HDRREC,        ALIAS=E01,   USAGE=A11,      ACTUAL=A11,         $
   FIELD=RECTYPE,       ALIAS=H,     USAGE=A1,       ACTUAL=A1,          $
   $     HDRVALUE, VALUE 'H'.                                            $
   FIELD=HDRDATA,       ALIAS=E03,   USAGE=A10,      ACTUAL=A10,         $
SEGNAME=DETRESEG,      SEGTYPE=S0,  PARENT=DUMMY,                        $
   GROUP=DETREC,        ALIAS=E04,   USAGE=A11,      ACTUAL=A11,         $
   FIELD=RECTYPE,       ALIAS=D,     USAGE=A1,       ACTUAL=A1,          $
   $     DTLVALUE, VALUE 'D'.                                            $
   FIELD=DTLDATA,       ALIAS=E06,   USAGE=A10,      ACTUAL=A10,         $
```

Maximum Number of Fields

Each 01-level record input can contain up to 4,000 fields. Any number of 01-level entries can be used as input to the translation, as long as no individual record exceeds this limit. Each occurrence of an OCCURS structure counts in this limit. If the limit is exceeded, an error message is printed and program execution stops. Note that this limitation applies to the translation and not the Master File generated.

Example: Maximum Number of Fields

Consider the following COBOL input description:

```
01  REC1.
    02  FLD1  OCCURS  5000  TIMES  PIC  X(1).
```

This input generates the following message:

```
FR011E - INTERNAL TABLE OVERFLOW
```

To bypass this limitation, edit the COBOL FD and temporarily reduce the total number of OCCURS. After execution, edit the Master File and restore the original number of occurrences.

Consider the following COBOL input description:

```
01  REC1.
    02  FLD1  OCCURS  2000  TIMES  PIC  X(1).
01  REC2.
    02  FLD2  OCCURS  2000  TIMES  PIC  X(1).
```

The following Master File, with multiple segments, is created from this input:

```
FILE=MAXRECS,                    SUFFIX=FIX,                          $
SEGNAME=DUMMY,        SEGTYPE=S0,                                     $
   FIELD=,             ALIAS=,      USAGE=A1,       ACTUAL=A1,       $
SEGNAME=REC1SEG,      SEGTYPE=S0,  PARENT=DUMMY,                      $
$  GROUP=REC1,         ALIAS=E01,   USAGE=A2000,    ACTUAL=A2000,    $
   FIELD=FLD1SEGPOSN,  ALIAS=E02,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E03,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E04,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E05,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E06,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E07,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E08,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E09,   USAGE=A208,     ACTUAL=A208,     $
SEGNAME=FLD1SEG,      SEGTYPE=S0,  PARENT=REC1SEG,
OCCURS=2000,          POSITION=FLD1SEGPOSN,                           $
   FIELD=FLD1,         ALIAS=E10,   USAGE=A1,       ACTUAL=A1,       $
SEGNAME=REC2SEG,      SEGTYPE=S0,  PARENT=DUMMY,                      $
$  GROUP=REC2,         ALIAS=E11,   USAGE=A2000,    ACTUAL=A2000,    $
   FIELD=FLD2SEGPOSN,  ALIAS=E12,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E13,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E14,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E15,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E16,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E17,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E18,   USAGE=A256,     ACTUAL=A256,     $
   FIELD=FILLER,       ALIAS=E19,   USAGE=A208,     ACTUAL=A208,     $
SEGNAME=FLD2SEG,      SEGTYPE=S0,  PARENT=REC2SEG,
OCCURS=2000,          POSITION=FLD2SEGPOSN,                           $
   FIELD=FLD2,         ALIAS=E20,   USAGE=A1,       ACTUAL=A1,       $
```

This description does not exceed the 4,000 field limit because each 01-level has fewer than 4,000 fields.

Year 2000

Because a COBOL FD cannot describe fields as dates, the translation cannot reliably determine which fields are dates and apply date formats to them. However, you can edit Master Files generated during synonym creation to add date formats and date defaults. These are respected in Information Builders software designed to take advantage of them.

COBOL FD Syntax Requirements

The COBOL FD translation requires a syntactically correct COBOL file description as input. While scanning the COBOL file description during synonym creation, a number of syntax checks are performed. If any of the COBOL statements are invalid, debugging messages are displayed, the program stops executing, and a Master File is not generated. Although the syntax check can identify a variety of errors, you should specify only valid COBOL file descriptions that compile successfully.

Reference: COBOL FD Syntax Guidelines

At a minimum, a valid COBOL file description must follow these guidelines:

- Level-numbers must be a one-digit or two-digit integer. Values can be 1 through 49, 66, 77, or 88.
- Level-numbers 01 and 77 must begin in Columns 8 through 11; all other level-numbers may begin in Columns 8 through 72.
- Comment lines are identified with an asterisk (*) in Column 7.
- Continuation lines are identified with a hyphen (-) in Column 7.
- The COBOL file description starts with a level-number 01 column and ends with the last column described.
- The COBOL field names are in uppercase characters.
- Other COBOL instructions are not permitted as input.
- Embedded tabs and blank lines are not permitted as input.
- Up to nine lines are supported in a multilevel value clause.

  Only one level is commented in the Master File.

Appendix E

Data Set Compression Exit: ZCOMP

The ZCOMP Exit applies to VSAM, ISAM, CISAM, RMS and other SAM type data structures, and to certain FIX structures.

Three functions plus a main entry point comprise the compression exit, collectively known as the ZCOMP Exit. The main entry point is ZCOMP which exports:

- ZCOMP0 for initial housekeeping.
- ZCOMP1 for the actual processing. It is designed so customers can access and decrypt coded fields, expand compressed records, and accomplish other specified data manipulations.
- ZCOMP2 is called to perform clean-up when the session is over.

Anchor storage is passed to ZCOMP0, ZCOMP1 and ZCOMP2 to maintain persistence and reentrancy.

The ZCOMP reference code is supplied, However, the actual DLL is expected to be user-written, and maintained to fit the specific need. The exit must be compiled and linked in accordance with standard user-written exit procedures.

There are no special Master File coding requirements to use ZCOMP. The SUFFIX value can be VSAM (for KSDS or ESDS files), CISAM, RMS, FIX (for sequential files), or PRIVATE (for file access through the GETPRV user exit).

In this appendix:

Invoking the ZCOMP Exit

The ZCOMP Exit DLL may be written in C, BAL, or any 3GL languages that follows the parameter passing conventions used in the reference samples and are compiled and linked as a proper DLL. The generated DLL must have an entry point that matches the physical name of the DLL. The default DLL name is ZCOMP.

After you write a ZCOMP user exit, depending upon the platform, you compile and link it as follows:

Platform	Compile/Link Mechanism
USS, UNIX, Windows, IBM i, and OpenVMS	GENCPGM. (See the *Stored Procedures Reference* manual for related information.)
z/OS	Build the DLL or link with VVSET using FOCCTL.DATA(GENFSAM) JCL
VM/ESA	Build the DLL or link with VVSET using GENFSAM EXEC

Note:

- z/OS linking with VVSET is an obsolete technique that is only supported for backward compatibility.
- The GENFSAM JCL and EXEC for mainframe are designed to link to both the ZCOMP and the GETPRV user exits at the same time. If you are only implementing one of these exits, the VM EXEC generates the correct linkage to the routine required, whereas the z/OS JCL does not. In the latter case, you must comment out the INCLUDE OBJECT statements in the GENFSAM member.
- On z/OS, the ZCOMP can be linked as re-entrant if you plan to use the USERWD parameter. For information about GENCPGM usage, see the *Stored Procedures Reference* manual.

The point in the process at which the ZCOMP Exit DLL is loaded depends on the platform and on whether linking is done with VVSET:

- Platforms using exits linked into VVSET receive the default ZCOMP exit load when a user session starts; it is unloaded on exit.
- Platforms using the exit as a DLL load only in response to an explicit request, such as SELECT or TABLE, which must be issued prior to data requests. The commands specify the data source, as in the following illustrations:

```
ENGINE VSAMX SET ZCOMP dllname
ENGINE CISAM SET ZCOMP dllname
ENGINE RMS SET ZCOMP dllname
```

Users can dynamically change the DLL during the session as needed. The current DLL unloads upon the change to the DLL in use and upon a user session exit.

To force an explicit DLL unload, use the SET command with a non-existent DLL name. The DLL location is defined by the IBICPG environment variable. The main DLL entry point function (which is required to be the same as the DLL name) provides the adapter with the addresses of the ZCOMP0, ZCOMP1 and ZCOMP2 functions.

The VSAM engine is used to process SUFFIX=PRIVATE and multi-segment SUFFIX=FIX files, as well as SUFFIX=VSAM files. The following command is used for FIX and PRIVATE:

```
ENGINE VSAMX SET ZCOMP
```

What Happens When ZCOMP is LOADED?

An adapter (as defined by the SUFFIX= value in the Master File) reads records. Upon each successful read, if the ZCOMP Exit DLL is loaded the adapter calls ZCOMP1 to do manipulations, such as data decompression.

The ZCOMP1 logic is responsible for determining what to do based upon the parameter information it receives. Typically, the DDNAME value is used to determine if the associated data source needs to be decompressed or otherwise manipulated.

- If neither decompression nor manipulation is needed, the user exit typically moves A(IRECLEN) to A(ORECLEN) and A(A(IREC)) to A(A(OREC)) and returns to the adapter with a zero A(STATCODE), thus acting like a pass through. For related information, see *ZCOMP1 BAL Parameter List* on page 2725.
- If decompression or another type of manipulation is required, it is the responsibility of the user exit to perform these tasks.

Example: Passing Records

Note that the following example simply passes records since decompression, or another type of manipulation, is application-specific.

After the user exit completes its processing, it returns either A(ORECLEN), A(A(OREC)), and a zero status code, or a non-zero status code that generates the following message:

```
(FOC1150) ZCOMP DECOMPRESS ERROR: status
```

This error terminates a TABLE request.

Example: ZCOMP C Source File (zcomp.c)

The ibisamp application includes a ZCOMP FOCEXEC that can be used to "exercise" this exit example. The .c and .h sources may be found in the equivalent of the EDAHOME/etc/src3gl directory for your platform.

```
/*-------------------------------------------------------------------
* Copyright (c) 2003 Information Builders, Inc.  All rights reserved.
*
* _Name_        ===> ZCOMP    C           _Opsys_    ===> CMS
* _Release_     ===> 72                   _Product_  ===> FOC
* _Description_ ===> Example of the dynamic ZCOMP exit functions
*               ===>
* _Notes_       ===> @MFSM_NOPROLOG@ source control tag
*-------------------------------------------------------------------*/

/*
  This zcomp sample is an arbitrary example in which we simply move
  an uncompressed input buffer and size to the output buffer and size.
  Effectively, this is a pass through of operation just passing the
  buffer along; no real decompression occurs in the sample. If this was
  coded as a real application, actual decompression would be done using
  a third party compression/decompression routine linked in the sample
  for the purpose of actual use in the ZCOMP1 exit.

  Typically this procedure would be built via gencpgm and IBICPG set.
  Once the sample exit has been built, it may be exercised to
  observe behavior using the zcomp.fex procedure.
*/

/* stdio needed for printf() lines, your code may not need stdio */
#include <stdio.h>
#include "zcomp.h"
void zcomp0 (
              long   *status     /* out: status, 0=OK                */
             ,char   *filename   /* in : ddname                      */
             ,char   *userid     /* in : userid or jobname           */
             ,long   *reserv1    /* reserved: NOT TO BE USED BY EXIT  */
             ,char  **reserv2    /* reserved: NOT TO BE USED BY EXIT  */
             ,long   *reserv3    /* reserved: NOT TO BE USED BY EXIT  */
             ,char  **reserv4    /* reserved: NOT TO BE USED BY EXIT  */
             ,long   *user_wk    /* i/o : work area for the user exit */
            )
     {
       /* Arbitrary to show behavior; would not be coded in a real application  */
       printf("ZCOMP0 called to do initial housekeeping\n");
       *status = 0;
     }
```

```
void zcomp1 (
              long   *status     /* out: status, 0=OK                    */
             ,char   *filename   /* in : ddname                          */
             ,char   *userid     /* in : userid or jobname               */
             ,long   *inlen      /* in : length of original record       */
             ,char  **inabuf     /* in : input record buffer             */
             ,long   *outlen     /* out : length of decoded record       */
             ,char  **outabuf    /* out : decoded output record buffer   */
             ,long   *user_wk    /* i/o : work area for the user exit    */
            )
     {
       /* Arbitrary to show behavior; would not be coded in a real application  */
       printf("ZCOMP1 called to do the actual processing\n");
       /* In a real application decompression steps would normally be done here */
       *outlen = *inlen;
       *outabuf = *inabuf;
       *status = 0;
     }

void zcomp2 (
              long  *user_wk  /* I/O work area for the user exit */
            )
     {
       /* Arbitrary to show behavior; would not be coded in a real application  */
       printf("ZCOMP2 called to perform clean-up, session is over\n");
     }

void zcomp(p_zcompep pzc)
    {
      /* Arbitrary to show behavior; would not be coded in a real application  */
      printf("ZCOMP called to initialize entry points for ZCOMP0, ZCOMP1 and
ZCOMP2\n");
      pzc->zcomp0 = zcomp0;
      pzc->zcomp1 = zcomp1;
      pzc->zcomp2 = zcomp2;
    }
```

Example: **ZCOMP Header File (zcomp.h)**

```
/*--------------------------------------------------------------------
* Copyright (c) 2003 Information Builders, Inc.  All rights reserved.
*
* _Name_        ===> ZCOMP    H           _Opsys_    ===> CMS
* _Release_     ===> 72                   _Product_  ===> FOC
* _Description_ ===> ZCOMP api to process compressed records
*               ===> read by the VSAM interface
* _Notes_       ===> Applicable to all files processed by VSAM int
*               ===> @MFSM_NOPROLOG@ source control tag
*--------------------------------------------------------------------*/
```

```
#ifndef ZCOMP_H
#define ZCOMP_H 1
/*
---------------------------------------------------------------------
    -perform the set-up in the user exit
---------------------------------------------------------------------
*/
typedef void  t_zcomp0 (
 long       *status     /* out: status = 0 if OK*/
,char       *filename   /* in : ddname            */
,char       *userid     /* in : userid or jobname */
,long       *reserv1    /* reserved: NOT TO BE USED BY THE EXIT */
,char      **reserv2    /* reserved: NOT TO BE USED BY THE EXIT */
,long       *reserv3    /* reserved: NOT TO BE USED BY THE EXIT */
,char      **reserv4    /* reserved: NOT TO BE USED BY THE EXIT */
,long       *user_wk    /* i/o : work area for the user exit */
);

/*
---------------------------------------------------------------------
    -decompress a record   and return the address of the new
     record and its length. The routine is responsible for allocating
     a buffer for the decompressed record.
---------------------------------------------------------------------
*/
typedef void  t_zcomp1 (
 long       *status     /* out: status = 0 if OK*/
,char       *filename   /* in : ddname            */
,char       *userid     /* in : userid or jobname */
,long       *inlen      /* in : length of original rec */
,char      **inabuf     /* in : a' input record        */
,long       *outlen     /* out : length of decoded  rec */
,char      **outabuf    /* out : a' decoded record      */
,long       *user_wk    /* i/o : work area for the user exit */
);
```

```
/*
-------------------------------------------------------------------
    -deallocate buffer used by zcomp1
     called at the end of focus session.
-------------------------------------------------------------------
*/
typedef void  t_zcomp2 (
  long       *user_wk    /* i/o : work area for the user exit */
);
/*
-------------------------------------------------------------------
    -definition of EPs exported by the dynamic zcomp exit
-------------------------------------------------------------------
*/
typedef struct s_zcompep
{
  t_zcomp0 *zcomp0; /* p' to zcomp0                */
  t_zcomp1 *zcomp1; /* p' to zcomp1                */
  t_zcomp2 *zcomp2; /* p' to zcomp2                */
} t_zcompep, *p_zcompep;


/*
   Main and only one entry point in the ZCOMP module (dll)
*/
#ifdef __cplusplus
extern "C" {
#endif
typedef void  t_zcomp(p_zcompep pzc);
t_zcomp zcomp;
#ifdef __cplusplus
} /* extern "C" */
#endif /* __cplusplus */
#endif
```

Reference: ZCOMP1 BAL Parameter List

Parameter	Description	Length and Format
A(STATCODE)*	Pointer to full word binary status code	4-byte integer
A(DDNAME)	Pointer to the 8-byte file name in use	8-byte character
A(USERID)	Reserved for future use	8-byte character
A(IRECLEN)	Pointer to length of original record	4-byte integer
A(A(IREC))	Pointer to pointer to original record	4-byte integer
A(ORECLEN)*	Pointer to length of revised record	4-byte integer

Parameter	Description	Length and Format
A(A(OREC))*	Pointer to pointer to revised record	4-byte integer
A(USERWD)**	Pointer to full word	4-byte integer

* The user supplies these parameters.

** This parameter can be used to anchor user storage for re-entrant processing.

Note:

- Never modify the original pointers, only the pointers or values that they point to.
- Upon entry to ZCOMP1, the ORECLEN and OREC parameters are NULL. It is the responsibility of the user to fill these in correctly.
- While processing the end of session, a call is placed to the ZCOMP2 entry point, which enables the user to do any other global cleanup that is required.
- The parameters returned by ZCOMP1 are not validated. It is the responsibility of the user routine to ensure that valid addresses and lengths are returned to the caller from ZCOMP1.
- Unpredictable results occur if incorrect parameters are passed back from the routine.

Appendix F

Dynamic Private User Exit

The Dynamic Private User Exit is an API that enables you to build and integrate a custom 3GL dynamic load library (DLL) routine to access data sources for which no standard adapter is available. This technique provides a mechanism by which companies using proprietary (or obscure) data sources can take advantage of WebFOCUS data access.

The terms GETPRV and SUFFIX=PRIVATE are often used to refer to this exit.

While reference code and build instructions for the exit are supplied as part of the GETPRV exit, users are expected to write and maintain the actual DLL to fit specific application needs, and compile and link the exit in accordance with the standard procedures for designing and building user written exits, as described in the *Stored ProceduresReference* manual.

To facilitate troubleshooting, customers (or VARs) are responsible for following good practices in coding their applications. If a problem is encountered it should be reproducible through small representative modifications to the sample. Since specific application code may contain proprietary data or methods that slow down the resolution process with unnecessary complexity, this method should be used to provide reproductions of all problems.

In this appendix:

FOCSAM and the GETPRV User Exit

FOCSAM is the underlying logic for accessing keyed and sequential data sources within Information Builders products using a variety of low-level physical retrieval modules. FOCSAM contains a user exit that can be invoked as an alternative to the standard low-level retrieval routines that are part of FOCSAM. The exit makes it possible to combine user-written code (as a DLL) devoid of any dependence on the internal structures used within IBI products with the logical retrieval functions of FOCSAM, such as record selection logic, treatment of missing records in multi-record files, JOINS between various types of files, etc.

An Access File with a MODNAME= *value* determines the specific physical DLL loadable library to be used for a given table. The DLL itself must have a matching entry point by the same name as dictated by IBI standards for user written subroutines. The DLL must be physically in the EDACONF user directory, or the IBICPG environment variable must be set to indicate an alternate (directory) location. Any additional entry points within the same module may also be accessed, however, if other modules with additional entry points are needed the Private User Exit module must include "loader" logic to make the entry points available. (Note that these are the same rules used for all customers writing loadable DLL subroutines, as described in the *Store Procedures Reference*, which you can consult for the finer points of physically building a proper DLL.)

Procedure: How to Use the Dynamic Private Exit User Exit

This is an overview of the steps for using the Dynamic Private User Exit:

1. Determine a record layout for the interaction between FOCSAM, the DLL, and any required keys. From the FOCSAM perspective this is the metadata layout within a given Master File. From the DLL perspective this is the structure of the record data being returned within one of the API pointers.
2. Create a Master File with metadata that matches the chosen record layout.
3. Create an Access File that specifies the DLL that will be used to access and return the record data.
4. Design and create the DLL that will perform the data access and return the record data.
5. Set the IBICPG environment variable to indicate the (directory) location of the DLL before server start up, or place the DLL in the EDACONF user directory.
6. Access data as if it were just another table (although the actual specifics of the metadata and DLL are more involved).

General Features of the GETPRV Exit

The general features of the exit are:

- The CONTEXT parameter, which supports reentrancy, reduces storage requirements, and enhances invocation performance.
- Support for multiple concurrent exit processors provided through an Access File where a specific user exit module can be named on a per Master File basis.
- Dynamic invocation of the user exit at execution time, without the need to recompile or re-link between major releases.

  Note: Since the exposed parts of the API are stable, the need for application code changes, re-compilation, or re-linking is unlikely between major releases. However, IBI reserves the right to make such changes and customers using the exits should safeguard their source code so that it will be available should such changes be required or should the application need to move to a new platform.
- Initialization and shutdown calls that allow the exit to perform initial and final housekeeping.
- Control and Read options:
  - Control options: (O) OPEN file, (R) OPEN request (position), (C) CLOSE, and (F) FIN (FOCUS FIN or WebFOCUS Agent Exit)
  - Read options: (S) Sequential Read, (G) Generic (GE) Read, and (E) Direct (EQ) Read options.
- Maintain multiple positions on the same file for recursive Joins.

Functional Requirements for Using the GETPRV Exit

Functionally, the exit is a substitute for retrieval calls typically used against key-sequenced VSAM data sources or any data source that can be represented as such a file. The exit need not deal with intra-record structures represented by OCCURS phrases, nor with the translation of FOCUS IF/WHERE conditions into lower-level criteria. Both of these functions are performed by the driving logic within FOCSAM.

The user-written code must conform to the following rules:

- The code must be able to obtain and return a record, given a full value of the key. The key is presumed to be unique (E Direct (EQ) Read) and only one access is attempted (no successive reads).

- The code must be able to obtain and return a record greater than or equal to a given value of the full or partial key (G Generic (GE) Read) for initial positioning and retrieval purposes. After the initial retrieval, access switches to a sequential read until a change in key is detected and the position/retrieve (generic read) logic is repeated for any successive keys (for example, IF COUNTRY EQ ENGLAND OR ITALY) else ends retrieval (CLOSE). It is assumed the records being access and retrieved are in sort order to allow proper positioning and detection of key changes. If the exit application code uses an API to do retrieval specifics, sort order specifics may be masked or irrelevant, but the need for sort order retrieval becomes obvious when used in the context of a plain text file such as used in the reference sample.
- The code must be able to obtain the next logical record, starting from the current position in the file (S Sequential Read). Successive sequential reads must return records in ascending sequence (bit by bit). No key or non-key information is passed except for subsequent sequential reads after a Generic (GE) Read on a secondary key. For simplicity (but not efficiency), the passed key information can also be ignored and the record returned as FOCSAM will be applying screening conditions also.
- Direct and Generic Reads that retrieve records must establish the starting position in the file for subsequent sequential reads. Direct reads that fail to retrieve the requested records need not establish these positions as only the initial request is attempted.
- For the open file request (O), the code must logically or physically open the file.
- If the logical end-of-file is reached as a result of a sequential read, an end-of-file signal must be returned by setting the record return length to zero. Subsequent sequential reads must return end-of-file signals rather than error indications, for example, when processing a JOIN. Error condition handling variable should not be used to indicate end-of-file.
- A close function must be provided that will result in the release of all resources and loss of position in the file, so that subsequent open requests will succeed.
- Successive close calls must be innocuous. For example, they must be prepared to handle conditions such as a close of a file that is already closed, without generating error conditions.
- A unique area may be obtained and maintained using the CONTEXT parameters on a per DDNAME basis.
- The code must be serially reusable and re-entrant. This is the norm for modern compilers and linkers, but on MVS this would be linking as AMODE 31.

Physical Implementation of the GETPRV Exit

The GETPRV exit is linked and called as a separate dynamically loadable module in compliance with the standard instructions for subroutine DLL construction and access. The process can be summarized as follows:

- Write the set EDAHOME program.
- Build and compile the 3-GL application with GENCPGM.
- Set IBICPG to the name of the directory where the application was built.
- Start the Server and issue an SQL or TABLE request.

Once the program is working properly, only the last two steps are necessary: set IBICPG and start the Server.

For details about subroutine DLL construction and use of the GENCPGM program, see the *Stored Procedures Reference* manual.

Only READ features of FOCSAM are supported.

Master File for Data Access With GETPRV

The Master File and attributes for data to be accessed using the GETPRV exit are the same as the attributes for any other FOCSAM data source, except that the SUFFIX value must be PRIVATE (no other value will invoke the exit), and a primary key must be specified as a GROUP with an ALIAS=KEY value. For example,

```
FILE=filename, SUFFIX=PRIVATE, $
 SEGNAME=ROOT, SEGTYPE=S0, $
  GROUP=keyname, ALIAS=KEY, USAGE=xx, ACTUAL=xx, $
   FIELD=fieldname1, ALIAS=aliasname1, USAGE=xx, ACTUAL=xx, $
   FIELD=fieldname2, ALIAS=aliasname2, USAGE=xx, ACTUAL=xx, $
```

Secondary keys are typically set up as follows:

```
FILE=filename, SUFFIX=PRIVATE, $
 SEGNAME=ROOT, SEGTYPE=S0, $
  GROUP=keyname, ALIAS=KEY, USAGExx, ACTUAL=xx, $
   FIELD=fieldname1, ALIAS=aliasname1, USAGE=xx, ACTUAL=xx, $
   FIELD=fieldname2, ALIAS=aliasname2, USAGE=xx, ACTUAL=xx, $
   FIELD=fieldname2, ALIAS=KEY1      , USAGE=xx, ACTUAL=xx, INDEX=I, $
   FIELD=fieldname2, ALIAS=KEY2      , USAGE=xx, ACTUAL=xx, INDEX=I, $
```

They may also be set up as follows:

```
FILE=filename, SUFFIX=PRIVATE, $
 SEGNAME=ROOT, SEGTYPE=S0, $
  GROUP=keyname , ALIAS=KEY , USAGE=xx, ACTUAL=xx, $
   FIELD=fieldname1,  ALIAS=aliasname1, USAGE=xx, ACTUAL=xx, $
   FIELD=fieldname2,  ALIAS=aliasname2, USAGE=xx, ACTUAL=xx, $
   GROUP=keyname    , ALIAS=KEY1      , USAGE=xx, ACTUAL=xx, INDEX=I, $
    FIELD=fieldname2, ALIAS=aliasname3, USAGE=xx, ACTUAL=xx, $
   GROUP=keyname    , ALIAS=KEY2      , USAGE=xx, ACTUAL=xx, INDEX=I, $
    FIELD=fieldname2, ALIAS=aliasname4, USAGE=xx, ACTUAL=xx, $
```

All standard rules for keys and discontiguous keys as described in the standard manuals for describing keyed sequential sources apply to the use of SUFFIX=PRIVATE and should also be consulted.

To reference a table, you must specify the physical name of the Master File (for example, mytable.mas) and it must be located in an application directory on the server application path. On MVS configurations using PDS for storing application files, the Master File must be in the PDS allocated to the DDNAME MASTER or be in a PDS for a given applications Master Files.

Access File for Data Access With GETPRV

An Access File is required. It provides the actual physical name of the private exit by specifying a MODNAME value. On platforms like UNIX, which use DLLs naming conventions such as lib*xxxx*.so. The *xxxx* portion is the value to specify as the MODNAME. For example,

```
MODNAME=pgmname,$
```

To reference a table, you must specify the physical name of the Access File (for example, mytable.acx) and it must be located in an application directory on the server's application path. On MVS configurations using PDS for storing application files (where there are no file extensions), the Access File must be in the PDS allocated to the DDNAME ACCESS or be in a PDS for a given applications Access Files.

Calling Parameters and Work Areas

A user-coded retrieval routine must be written as a standalone DLL program. There are few limitations to the program other than standard C rules and a program name length of 8. It is advisable to write the program in a language that fully supports dynamic memory allocation, such as ASSEMBLER or C. Languages like COBOL can also be used, but with certain limitations in the ability to access extended parameters and set permanent address memory on some platforms.

Note that non-char parameter sizes are stated in terms of longs and pointers: 4 bytes each for 32-bit applications and 8 bytes each for 64-bit applications.

The basic function of a user-coded routine is to provide calling parameters for opening, positioning, reading, and closing in order to populate work areas with records to be passed back. For details, see *Calling Parameters* on page 2733 and *Work Area Control Block Parameters* on page 2735. A sample is also provided for your use.

Reference: Calling Parameters

The parameter list is as follows:

Parameter	Definition
NCHAR	Long, posted by the exit; a positive number indicates the length in bytes of the record obtained and being returned; a zero indicates no record retrieved or end-of-file has been reached (there is no functional difference). NCHAR must be set to a non-zero for every successful read that returns data. Used by read options (S), (E), and (G).
DDN	Eight-byte character argument, posted by FOCSAM. The value corresponds to the table name for the request. For example, TABLE FILE MYTABLE results in DDN (also known as DDNAME) of MYTABLE, left-justified and blank-padded. Used for all options except (F). Do not modify this parameter. The value is normally used to indicate the desired file to open unless the extended information parameter for filename is available and non-blank.
ABUF	Pointer, posted by the exit; the pointer is the absolute address of the record obtained and being returned to FOCSAM. Used with all read options (S), (E), and (G). Do not modify this pointer. Instead, modify the location addressed by the pointer.
RC	Long, return code posted by the exit. Zero indicates no error: non-zero indicates some type of error. Specific non zero values do not activate specific error handlers, there is only one handler and it returns a FOC1143 message. Specific text can be passed back in the extended information RETTEXT parameter so that the generic FOC1143 message and specific text may be displayed. Used with all options.

Parameter	Definition
KEY	String of 255 bytes posted by FOCSAM, containing the full value of the key for direct or partial keys for generic reads. Not significant for sequential reads. The key value is left justified and limited to 255 bytes. Note that the exit must know the exact key length and its format. For simple cases with a single key, it may suffice to hard-code the value, else information may also be obtained from the extended information parameters. KEY is used with options (E) and (G). Do not modify this parameter.
OPT	Four-byte character argument, posted by FOCSAM; the value indicates the type of request. Do not modify this parameter. The options are as follows: READ OPTIONS 'S' = Sequential Read. 'E' = Direct Read (EQ). 'G' = Generic Read (GE). CONTROL OPTIONS 'O' = OPEN File. 'R' = OPEN Request (position) used for recursive JOINs. 'C' = CLOSE File. 'F' = FIN (FOCUS FIN) - final housekeeping step should be done. These control and read arguments must include three trailing blanks. The standard C header file for the exit (getprv.h) may contain additional references for IBI use only and are not supported for customer purposes.
* (getprv_info)	Pointer, posted by FOCSAM, pointing to the extended information work area structure (getprv_info). For details, see *Work Area Control Block Parameters* on page 2735 described below. Do not modify this parameter.

Note: The parameters passed to the exit are not delimited by having the high-order bit set on in the last parameter. Make sure that your program does not scan for this high-order bit.

Reference: **Work Area Control Block Parameters**

The parameter list for the work area control block is an extension of the basic calling parameters. The list includes:

Parameter	Definition
EYECATCH	An eight-byte string, posted by FOCSAM. The value always contains the literal 'PRIVATE '.
PFMCB	Pointer, posted by the exit. The value is generally set during option (O) processing by the exit program and returned unchanged by subsequent FOCSAM calls. This parameter is generally used to point to the dynamic work areas used to maintain reentrancy.
PFACB	Pointer, posted by the exit. The value is generally set during option (O) processing by the exit program and returned unchanged by subsequent FOCSAM calls for options (C), (R), (E), (G), and (S) and is unique by DDN. This is generally used as a physical file context. This parameter is not valid for option (F).
PFRPL	Pointer, posted by the exit. The value is generally set during option (R) processing by the exit program and returned unchanged by subsequent FOCSAM calls for options (E), (G), and (S) and is unique for each view within the above PFACB parameter. This is generally used for logical file context. This parameter is not valid for option (F).
KEYLEN_FIL	Long, posted by FOCSAM. The value is set during option (O) processing by the exit and contains the whole key length for the file.
KEYLEN_REQ	Long, posted by FOCSAM. The value is set during options (G) and (E) processing by the exit and contains the actual key length for a direct read since it may be less than KEYLENF in a multi-field group key.

Parameter	Definition
ERRTEXT	Pointer, posted by the exit. The value is the absolute starting address of an error message to be passed back. A FOC1143 and this message are displayed if the RC parameter returned by the exit is non-zero.
LERRTEXT	Long, posted by the exit. The value must contain the length of the above ERRTEXT message.

INDEX	Full-word binary integer, posted by FOCSAM, based on the Master Files metadata. This option contains the index # by which to access the file.
	`0`
	Is the primary key in Master File, where ALIAS=KEY or DKEY.
	`1, 2, ...`
	Are secondary indexes in the Master File, where:
	`ALIAS= KEY1, KEY2 through KEYn and INDEX=I`
	or
	`ALIAS= DKEY1, DKEY2 through DKEYn and INDEX=I`

RESERVED1	Pointer, reserved for IBI use.
USERID	String of 8 bytes, posted by FOCSAM. The value is the user ID accessing the exit.
TRACENUM	Long, posted by the exit. The value must contain the number of lines in the TRACEARR pointer.
TRACEARR	Pointer, posted by the exit. The value is the absolute starting address of trace lines to be passed back and inserted into standard trace facilities.
RESERVED2, RESERVED3 and RESERVE4D	Set of three pointers, reserved for IBI use.

Parameter	Definition

FILENAME	Pointer, posted by FOCSAM. The location of this pointer contains a file name if a FILEDEF or DDNAME has been issued for the table name. For example:
	`FILEDEF MYTABLE DISK /home/ralph/myotherdata.ftm`
	`TABLE FILE MYTABLE`
	`...`
	`END`
	
	The pointer contains */home/ralph/myotherdata.ftm*, and may be used to as an actual file to open or a flag to indicate a file or a feature.

INDEXNAM	String of 8 bytes, posted by FOCSAM. The value is the ALIAS of secondary keys when access is via a secondary key. Only one secondary key may access a given request, although the secondary key may be a GROUP consisting of multiple fields.
RESERVED5 and RESERVED6	Set of two longs, reserved for IBI use.
RESERVED7	Pointer, reserved for IBI use.

Example: **Sample C Header File Implementation for Calling and Work Area Control Block Parameters**

For your convenience, the following file (getprv.h for SUFFIX=PRIVATE) is provided for inclusion in your C program. It contains both calling and work area control block parameters.

If you choose not to include this file, you are required to write and include your own header file in your C program.

```
typedef struct getprv_inf_s {
char        eye[8];      /* I: Eye Catcher "PRIVATE "                  */
void        *pfmcb;      /* O: Pointer to handle for getprv            */
                         /*    Set up by user at first option O        */
                         /* I: Pointer to handle for getprv            */
                         /*    Passed to user by all other calls       */
void        *pfacb;      /* O: Pointer to handle for file              */
                         /*    Set up by user at option O               */
                         /* I: Pointer to handle for file               */
                         /*    Passed to user at option C,R,E,G,S       */
void        *pfrpl;      /* O: Pointer to handle for request           */
                         /*    Set up by user at option R               */
                         /* I: Pointer to handle for request           */
                         /*    Passed to user at option E,G,S           */

long        keylen_fil;  /* I: Key length (whole) for the file          */
                         /*    Used at option O                         */
long        keylen_req;  /* I: Key length for the direct read request    */
                        /*     Used at direct read options G,E          */
char        *rettext;    /* O: Native db error msg text                   */
long        lrettext;    /* O: Length of native db error msg text          */
long        index;       /* I: Index # by which to access file:            */
                         /*   0 = Primary Key, (usually as a GROUP=)      */
                         /*    coded in MFD as ALIAS= KEY or DKEY         */
                         /*   1, 2, ... = Secondary Indexes,              */
                         /*     coded in MFD as ALIAS = KEYn and INDEX=I  */
                         /*                  or ALIAS = DKEYn and INDEX=I */

void*      reserved1;  /* IBI Use Only */
char       userid[8];  /* User ID                                        */
long       tracenum;   /* Number of 0 terminated trace lines           */
char        *tracearr;   /* Array of trace lines                                  */
void      *reserved2;  /* IBI Use Only */
void*      reserved3;  /* IBI Use Only */
char      *reserved4;  /* IBI Use Only */
char const *filename;   /* Full path name for UNIX, Windows, OpenVMS, OS400 and
USS. */

char        indexnam[8]; /* For secondary indexes, ALIAS name from MFD otherwise  */
                         /* blank. Populated prior to O (Open) option in FOCUS 7.2*/
                         /* and up. Field in MFD must have INDEX=I                 */
long        reserved5;   /* IBI Use Only */
long        reserved6;   /* IBI Use Only */
void*       reserved7;   /* IBI Use Only */
} getprv_inf_t ;
```

```
typedef void getprv_t (
 long       *nchar      /* O: Length of data record read       */
                        /*      0 = EOF or no record found     */
                        /*    Used in all read options S,E,G   */
,char       *ddn        /* I: DDName to read                   */
                        /*    Used in all options except F     */
,char       **abuf      /* O: Address buffer to return records */
                        /*     used in all read options S,E,G  */
,long       *rc         /* O: Return code. 0 if ok else non 0  */
                        /*    Used in all options              */
,char       *key        /* I: Key value for read               */
                        /*    Used in read options E and G     */

,char       *opt        /* I: Read option :                    */
                        /*    S   Sequential read              */
                        /*    G   GE read                      */
                        /*    E   EQ read                      */
                        /*    Control options                  */
                        /*    O   Open file                    */
                        /*    R   Open request (position)      */
                        /*    C   Close file                   */
                        /*    F   FIN of focus                 */
,getprv_inf_t  *        /* in/out for extended info above      */
);
```

Example: **Sample for SUFFIX=PRIVATE (getprv.c)**

Although the full sample (which is provided with the product) is too long to include in this document, the included section describes the requirements and steps for building and testing. The actual sample includes extensive annotation to serve as a model.

❑ Ensure that you have the following required sample files.

File	Description
getprv.c	The sample itself.
getprv.h	The h file used by the sample.
getprv.mas	Master file used by test request that call the sample.
getprv.acx	Access file used by test request that call the sample.
getprv.ftm	Data file used by test request that call the sample.
getprv.fex	Focexec test request that calls the table (.mas and .acx) used by the sample.

If you do not already have the Master File, Access File, data or procedure (focexec), the getprv.c file explains how to create them. If you do not already have the getprv.c and getprv.h files contact Customer Support.

- Other requirements are access to an installed and configured server, gencpgm (part of server software, and an execution tool such as a server configuration edastart, rdaapp or the Web Console. Note that genpgm must be from a 7.1.1 or higher release, but the programs it generates will work with any 5x or higher server.

  Set the EDAHOME environment variable to allow gencpgm to operate properly on the following platforms:

Platform	Command
UNIX, USS and OS400	`export EDAHOME=/home/iadmin/ibi/srv71/home`
OpenVMS	`DEFINE EDAHOME IWAY:[IADMIN.IBI.SRV71.HOME]`
Windows	`set EDAHOME=C:\ibi\srv71\home`

  Using the appropriate platform specific path for gencpgm, build the DLL. For example on UNIX:

```
$EDAHOME/bin/gencpgm.sh -m cpgm getprv.c
```

  For additional information about gencpgm usage and building DLLs if needed, see the *Stored Procedures Reference*.

- Set the IBICPG environment variable (which tells the Server where to find DLLs) to the name of the directory where the DLLs reside (typically the current directory), using the conventions for the following platforms:

Platform	Command
UNIX, USS and OS400	`export IBICPG='pwd'`
OpenVMS	`DEFINE IBICPG IWAY:[IWAY]`
Windows	`set IBICPG=C:\ibi`

Alternatively, you can use the EDACONF user directory as the working directory, in which case the DLLs will be picked up automatically.

- Execute the request using edastart -t", "edastart -x" or a front-end tool such as the web console or rdaapp. If exercising via a front-end tool, the DLL must be in the EDACONF user directory or the IBICPG variable must be set before server start up.

  To exercise with edastart x use an appropriate platform specific path. For instance:

  ```
  /home/iadmin/ibi/srv71/ffs/bin/edastart -x ex getprv
  ```

- After the initial test returns data (Sequential Read), modify the test and add various IF or WHERE statements to observe and exercise the Generic Read (G) and Direct Read (E) behaviors. Note that the sample has a number of printf statements so one can follow behavior of the sample. In a real application these statements would be deleted, commented out, or ifdefed.

Appendix G

Validation for Special Characters and Reserved Words

When creating a synonym, if you choose the *Validate* check box, the server adjusts special characters and checks for reserved words.

In this appendix:

Validation for Special Characters

If you choose the *Validate* check box while creating a synonym the following special characters are converted to underscores:

Character	Description	Character	Description
-	dash	%	percent
	space	|	vertical bar
\	backslash	&	ampersand
/	forward slash	(	open parenthesis
,	comma	)	closed parenthesis
$	dollar sign	|	vertical bar
.	period	<	less than
;	semicolon	>	greater than
+	plus sign	'	apostrophe
*	asterisk	=	equal
.	period	"	double quote

Validation for Reserved Words

If you choose the *Validate* check box while creating a synonym reserved words are checked for. The following reserved words cannot be used as names in the created synonym:

Reserved Word
ALIAS, ALL, ALTER, AND, ANY, AS, ASC, AUTHORIZATION, AVG
BEGIN, BETWEEN, BINARY, BIT, BOTH, BY
CALL, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, COALESCE, COLFETCH, COMMIT, CONCAT, CONNECT, CORRESPONDING, COUNT, CREATE, CROSS, CURRENT, CURSOR
DATABASE, DATE, DATETIME, DAY, DAYS, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DO, DOUBLE, DROP
ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXECUTE, EXISTS, EXPLAIN
FETCH, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION
GET, GRANT, GRAPHIC, GROUP
HAVING, HOUR, HOURS
IF, IMAGE, INCLUDE, IN, INDEX, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS, ITERAT
JOIN
KEY
LEADING, LEAVE, LEFT, LIKE, LOCK, LOGICAL, LONG, LOOP
MAX, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MIN, MINUTE, MINUTES, MONEY, MONTH, MONTHS
NATURAL, NOT, NULL, NULLIF, NUMBER, NUMERIC
OF, ON, ONLY, OPEN, OPTIMIZE, OPTION, OR, ORDER, OUT, OUTER
PACKAGE, PERCENT, PLAN, PRECISION, PREPARE, PRIMARY, PROCEDURE, PROGRAM, PURGE
QUERYNO

Reserved Word
RAW, REAL, REFERENCES, REPEAT, RESET, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROW, ROWS
SCHEMA, SECOND, SECONDS, SELECT, SERIAL, SET, SMALLFLOAT, SMALLINT, SOME, STOGROUP, SUM, SYNONYM, SYSNAME
TABLE, TABLESPACE, TEXT, THEN, TIME, TIMESTAMP, TO, TRAILING, TRUNCATE
UNION, UNIQUE, UNTIL, UPDATE, USER, USER_TYPE_NAME, USING
VALUES, VARBINARY, VARCHAR, VARGRAPHIC, VARYING, VIEW
WHEN, WHERE, WHILE, WITH, WORK
YEAR, YEARS

Index

B

c

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at *Sarah_Buccellato@ibi.com.*

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at *Frances_Gambino@ibi.com.*

Information Builders

WebFOCUS

Adapter Administration

WebFOCUS Reporting Server Release 8206

DataMigrator Server Release 7710

DN4501040.0619

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898