## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>7</td>
</tr>
<tr>
<td>Documentation Conventions</td>
<td>7</td>
</tr>
<tr>
<td>Related Publications</td>
<td>8</td>
</tr>
<tr>
<td>Customer Support</td>
<td>9</td>
</tr>
<tr>
<td>Information You Should Have</td>
<td>9</td>
</tr>
<tr>
<td>User Feedback</td>
<td>10</td>
</tr>
<tr>
<td>Information Builders Consulting and Training</td>
<td>10</td>
</tr>
<tr>
<td><strong>1. Introducing the Extender for Db2</strong></td>
<td>13</td>
</tr>
<tr>
<td>What Is the Extender for Db2?</td>
<td>13</td>
</tr>
<tr>
<td>Extender for Db2 Environment</td>
<td>13</td>
</tr>
<tr>
<td>Features of the Extender for Db2</td>
<td>15</td>
</tr>
<tr>
<td>Functional Overview of the Extender for Db2</td>
<td>16</td>
</tr>
<tr>
<td>Step 1. Intercepting Dynamic SQL Calls</td>
<td>16</td>
</tr>
<tr>
<td>Step 2. Determining Table Location</td>
<td>17</td>
</tr>
<tr>
<td>Determining Object Destination Logic</td>
<td>18</td>
</tr>
<tr>
<td>Step 3. Constructing and Dispatching Requests</td>
<td>20</td>
</tr>
<tr>
<td>Step 4. Returning Data to the Application</td>
<td>20</td>
</tr>
<tr>
<td><strong>2. Installing the Extender for Db2 on z/OS</strong></td>
<td>23</td>
</tr>
<tr>
<td>Installation Requirements</td>
<td>23</td>
</tr>
<tr>
<td>Hardware Requirements</td>
<td>23</td>
</tr>
<tr>
<td>Software Requirements</td>
<td>23</td>
</tr>
<tr>
<td>Disk Requirements</td>
<td>24</td>
</tr>
<tr>
<td>System Requirements</td>
<td>24</td>
</tr>
<tr>
<td>Pre-Installation Issues</td>
<td>24</td>
</tr>
<tr>
<td>Call Attach Facility</td>
<td>25</td>
</tr>
<tr>
<td>Explicit LOAD</td>
<td>25</td>
</tr>
<tr>
<td>Link-edit of DSNALI</td>
<td>26</td>
</tr>
<tr>
<td>DSN Command Processor Facility</td>
<td>26</td>
</tr>
<tr>
<td>Installation Worksheet for the Extender for Db2</td>
<td>26</td>
</tr>
<tr>
<td>Installation Procedure</td>
<td>28</td>
</tr>
<tr>
<td>Step 1. Unload Two Data Sets From the Tape</td>
<td>28</td>
</tr>
</tbody>
</table>
Step 2. Copy Extender Entry Points ............................................... 29
Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional) ........ 30
Step 4. Configure the Parameter File EDAPARMS (Optional) ......................... 31
Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional) .......... 31
   Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your
   Application (Optional) .......................................................... 32
Step 6. Prepare Run-time Allocation Streams ........................................... 33
   Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI .................. 33
   Sample CLIST .................................................................. 33
   Sample JCL ...................................................................... 35
   Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI ..................... 38
   Step 6.3. DSN Command Processor (TSO Attach) ..................................... 40
Installing the Extender for Db2 Without Db2 ............................................. 43

3. Configuring the EDAPARMS File ......................................................... 45
   Overview of the EDAPARMS File .................................................. 45
   Creating the EDAPARMS File ...................................................... 46
   Using the EDAPARMS File .......................................................... 49

4. Using the Extender for Db2 Administrative Operations ............................... 51
   Security ........................................................................ 51
     Using the EDA IMMEDIATE Command ........................................ 51
     Supported Types of Security ...................................................... 52
     Client Security .................................................................. 53
   Tracing ............................................................................. 53
     QXUDUMP Data Set File Allocation ............................................ 54
     Error Message Formatting Facility (DSNTIAR) .............................. 54

5. Using the Extender for Db2 and SQL ..................................................... 55
   Table Naming Conventions .......................................................... 55
     Fully-Qualified Tables ............................................................. 55
     Partially-Qualified Tables .......................................................... 56
     Product Work Tables ............................................................... 57
   SQL Translation .................................................................... 57
     Column Name Resolution ......................................................... 58
Alternate Column Names ......................................................... 58
Dynamically Defined Virtual Fields ................................................. 58
Answer Set Generation .......................................................... 58
  What Is a Cartesian Product? ................................................ 58
  What Does This Mean to You? ............................................... 59
  Answer Set Generation Logic ................................................ 59
  What to Look For .................................................................. 60
Additional Features for SQL Translation Services ................................ 60
SQL Translation Services Limitations .............................................. 61

6. Using the Extender for Db2 and Db2 SQL ........................................ 63
  General Considerations .............................................................. 63
  Data Conversion ................................................................ 66
  Error Handling .................................................................. 67
  Parameter Marker Support ....................................................... 68
Discrepancies Between the Server and Db2 SQL ........................................ 68
  Db2 Non-ANSI Compliant SQL Requests ........................................ 69
Answer Set Displays ............................................................. 70
Db2 Error Codes (SQLCODEs) .................................................... 70
Enhancements to Db2 ............................................................... 71
SAA CPI Functionality Checklist ........................................................ 75
Extender for Db2 Cross-reference Table ................................................. 79

7. Using Extender for Db2 Application Implementations ......................... 83
  Using QMF .......................................................................... 83
    Run-time CLIST ................................................................. 83
    The SQL SELECT Statement ...................................................... 85
    The Data Returned to QMF ....................................................... 85
  Using the Rocket Compiler for QMF ................................................ 86
    Prerequisites ................................................................... 86
    Installing the Rocket Compiler for QMF ........................................ 86
    Installing the Extender for Db2 ................................................ 87
  Using COBOL ....................................................................... 89
    Sample COBOL2 Program ....................................................... 89
Preface

This documentation describes how to use client Db2 applications on z/OS platforms (such as QMF, DXT, AS, DIS, or Language Access applications) to access remote data sources through servers.

How This Manual Is Organized

This manual includes the following chapters:

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introducing the Extender for Db2</td>
</tr>
<tr>
<td>2</td>
<td>Installing the Extender for Db2 on z/OS</td>
</tr>
<tr>
<td>3</td>
<td>Configuring the EDAPARMS File</td>
</tr>
<tr>
<td>4</td>
<td>Using the Extender for Db2 Administrative Operations</td>
</tr>
<tr>
<td>5</td>
<td>Using the Extender for Db2 and SQL</td>
</tr>
<tr>
<td>6</td>
<td>Using the Extender for Db2 and Db2 SQL</td>
</tr>
<tr>
<td>7</td>
<td>Using Extender for Db2 Application Implementations</td>
</tr>
<tr>
<td>A</td>
<td>Extender Error Messages and Codes</td>
</tr>
<tr>
<td>B</td>
<td>Connecting to Multiple Servers</td>
</tr>
</tbody>
</table>

Documentation Conventions

The following table describes the documentation conventions that are used in this manual.
## Convention

<table>
<thead>
<tr>
<th><strong>Convention</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>THIS TYPEFACE</strong> or <strong>this typeface</strong></td>
<td>Denotes syntax that you must enter exactly as shown.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Represents a placeholder (or variable) in syntax for a value that you or the system must supply.</td>
</tr>
<tr>
<td><strong>underscore</strong></td>
<td>Indicates a default setting.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Key + Key</strong></td>
<td>Indicates keys that you must press simultaneously.</td>
</tr>
<tr>
<td><strong>{  }</strong></td>
<td>Indicates two or three choices. Type one of them, not the braces.</td>
</tr>
<tr>
<td><strong>[  ]</strong></td>
<td>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.</td>
</tr>
<tr>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td>Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).</td>
</tr>
<tr>
<td><strong>.</strong></td>
<td>Indicates that there are (or could be) intervening or additional commands.</td>
</tr>
</tbody>
</table>

---

### Related Publications

Visit our Technical Content Library, [http://documentation.informationbuilders.com](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. Access Focal Point at 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 Services (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 software configuration:
  - The Server Software version and release. You can find your server version and release using the Version option in the Web Console.
    
    **Note:** the MVS and VM servers do not use the Web Console.
  - The communications protocol (for example, TCP/IP or LU6.2), 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?

Provide 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. Please use the Reader Comments form at the end of this document to communicate your feedback to us or to suggest changes that will support improvements to our documentation. You can also contact us through our website, http://documentation.informationbuilders.com/connections.asp.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Our Education Department offers a wide variety of training courses for the server 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.
Introducing the Extender for Db2

This section provides an overview of the Extender for Db2. It also describes the features and summarizes the functions of the Extender for Db2.

In the remainder of this manual, the Extender for Db2 will be referred to as the Extender for Db2 or, simply, the Extender.

In this chapter:
- What Is the Extender for Db2?
- Features of the Extender for Db2
- Functional Overview of the Extender for Db2

What Is the Extender for Db2?

The Extender for Db2 is a member of the Server family of products. The Extender for Db2 provides client Db2 applications on z/OS platforms (such as QMF, DXT, AS, DIS, or Language Access applications) with transparent access to remote data sources through servers. These data sources can be either relational tables and views or non-relational files from any of the data adapters supported by the server. The Extender for Db2 provides this access through Application Services and Network Services.

- Application Services provides a standard application programming interface through which client applications, including Db2 applications, send requests to servers and receive data in return. There is no need for client applications to be programmed to the details of the underlying operating systems, networks, or data sources of the server network.

- Network Services provides communications between connected client/server environments. It shields client applications from details associated with sending or receiving information across proprietary communications networks. Network Services on the client is comprised of the Communications Subsystem/3 (CS/3), supporting the TCP/IP protocol.

Extender for Db2 Environment

The following components make up a server environment with a Db2 application querying data sources:

- A Db2 application is an application that accesses data in a local Db2 subsystem.
The Extender for Db2 provides access to a server. It takes incoming SQL statements from a Db2 application and either passes them to the local Db2 subsystem, or converts them into appropriate server commands.

The Application Service receives the API calls and passes the request to the server in an appropriate form.

Network Services provides the communications interface for transporting requests to the server and for receiving returned data.

The Server processes the request and passes the resulting information back to the client.

This figure illustrates how these components fit into a Db2-Server environment.

The Db2-Server environment is comprised of the user application (Db2 application products, such as QMF and DXT, 3GL programs, or SQL tools), the Extender for Db2, and the server. It runs in its own address space. The server also runs in its own separate address space.
As shown in the figure above, the Db2 application can query data residing in the same Db2 subsystem in one of two ways:

- Through the Extender for Db2 and the server.
- Through the local Db2 subsystem via the Extender for Db2, bypassing the server.

The path on which the originating SQL request is directed (local Db2 or server) depends on the proper parsing of the table names.

**Features of the Extender for Db2**

The Extender for Db2 provides:

- Support of all SQL functions related to static or dynamic SQL directed at the local Db2 subsystem.
  
  **Note:** Static SQL calls require precompiling, binding from a Database Request Module (DBRM) to a static plan, and executing a load module through a static plan. Dynamic SQL calls are constructed and prepared into an executable object that contains access paths to Db2 databases. With dynamic SQL calls, the SQL request string is available to the Extender for Db2.

- Support of dynamic SQL functions for SELECT statements directed at the server. The dynamic SQL functions supported for SELECT statements are: PREPARE, DESCRIBE, OPEN, FETCH, and CLOSE.

- Support of dynamic SQL functions for non-SELECT statements directed at the server. The dynamic SQL functions supported for non-SELECT statements are: PREPARE, EXECUTE, and EXECUTE IMMEDIATE. The list of supported non-SELECT statements is limited to the functionality (Release level) of the server being utilized.

- Support of general purpose SQL statements (for example, COMMIT, ROLLBACK, and CONNECT).

- Operation using either the IBM Call-Attach Facility (CAF) or the IBM DSN Command Processor as application or call-level interfaces.

- Transparent interface.

Additional features include:

- **EDAPARMS.** This parameter file enables you to set the default error SQLCODE, to specify a default server for unresolved partially-qualified tables, to define worktables, and to enable only standard Db2 operation.
Explicit or implicit SQL CONNECT verbs. Enables the application to explicitly or implicitly CONNECT to a server or a local Db2 subsystem.

Functional Overview of the Extender for Db2

The Extender for Db2 operates as follows. It:

1. **Step 1. Intercepting Dynamic SQL Calls** on page 16
2. **Step 2. Determining Table Location** on page 17
3. **Step 3. Constructing and Dispatching Requests** on page 20
4. **Step 4. Returning Data to the Application** on page 20

**Step 1. Intercepting Dynamic SQL Calls**

Db2 applications communicate with their local Db2 subsystems using calls to interface modules. The Db2 application invokes a Db2 entry point pertinent to the communications mode used by the application. The table below describes the standard Db2 database entry points and their corresponding mode of database communication:

<table>
<thead>
<tr>
<th>Database Entry Point</th>
<th>Mode of Database Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNALI</td>
<td>Db2 Call Attach Facility</td>
</tr>
<tr>
<td>DSNELI</td>
<td>Db2 DSN Command Processor (Db2 TSO Attach Facility)</td>
</tr>
<tr>
<td>DSNTIAR</td>
<td>Db2 Error Message Formatting Facility</td>
</tr>
</tbody>
</table>

The Extender provides database entry points with names equivalent to the above standard names. Since the Extender resides between the application and the local Db2 subsystem, database calls made to the local database are intercepted by the Extender entry points.

The Extender for Db2 provides a Db2-like interface to the applications. Depending upon the application’s standard mode of communication with the local Db2 subsystem, it invokes the Extender entry point instead of the standard database entry point. Four common modes of Db2 entry are supported:

- **Call Attach Facility (CAF).** The application program invokes an Extender for Db2 DSNALI (DSNHLI2) replacement. In this mode, the application program can load this entry point dynamically in its own address space before the entry point’s invocation.
The Extender for Db2 uses table names to determine the location of the table in an SQL request. Within an SQL request, a user can specify access to a server data source or a local Db2 subsystem table by using table naming conventions. Table names are categorized as either fully-qualified (three-part) names, or partially-qualified (two-part or one-part) names. For more information on table naming conventions, see Using the Extender for Db2 and SQL on page 55.

When the Extender for Db2 parses the user’s SQL request, it determines whether to dispatch the request to the local Db2 subsystem or to accessible databases residing under a server. This determination is transparent to the user.

After the destination is determined for each table, the Extender for Db2 determines if every table in the SQL statement is consistently destined for either the local Db2 subsystem or a server:

- If every table is consistently destined for a local Db2 subsystem, then the request is simply passed to the Db2 DSNALI module.

- If every table is consistently destined for a server, then the appropriate commands are constructed and a communications dialog is established to dispatch the request to a server. (See Step 3. Constructing and Dispatching Requests on page 20 for more information about constructing local requests.)

- If the SQL statement references tables that are destined for both the local Db2 subsystem and a server, that request is rejected and the Extender for Db2 generates an error message.
The Extender for Db2 does not support SQL SELECT statements that contain tables from mixed destinations (such as a local subsystem table and a server table). This type of SQL request is rejected and an error message is generated.

To join tables from the local Db2 or SQL subsystem with any server-accessible database tables, define appropriate Master Files and Access Files for those Db2 tables under a common Hub Server.

**Determining Object Destination Logic**

For each SQL request, the Extender for Db2 first determines the destination for each table (either the local Db2 subsystem or a server). To do this, the Extender for Db2 uses the decision tree summarized here and illustrated graphically in the following flowchart.

1. Is the referenced table a:
   - Fully-qualified table name?
     Then go to Step 2.
   - Partially-qualified table name?
     Then go to Step 3.

2. Does the three-part name reference a:
   - Valid server?
     Then go to Step 5.
   - Undefined server?
     Then send request to the local Db2 subsystem.

3. Is the partially-qualified table name:
   - Defined in EDAPARMS under the WORKTABLE keyword?
     Then send request to the local Db2 subsystem.
   - Not defined in EDAPARMS?
     Then go to Step 4.

4. Was an:
   - SQL CONNECT TO issued?
     If the referenced server is a connected server, go to Step 5.
     If the server is not a connected Server, send the request to the local Db2 subsystem.
SQL CONNECT TO not issued?

If a default server was specified in the EDAPARMS file, go to Step 5.
Otherwise, send the request to the local database.

5. Was the requested option:
   - Supported by the server?
     Then send request to the server.
   - Not supported by the server?
     Then the Extender for Db2 supplies the appropriate error message.

This flowchart illustrates how the table destination is determined.
Step 3. Constructing and Dispatching Requests

Once it determines where the SQL request is to be dispatched, the Extender for Db2 constructs a request, depending on whether the request is destined for the local Db2 subsystem or the server.

- For a local Db2 request, it dispatches the original request to the Db2 DSNALI or DSNHLI load module.
- For a server request, it builds server calls around the request and establishes a communications dialog with the server.

Step 4. Returning Data to the Application

When data is returned to the Extender for Db2 from the local Db2 subsystem or the server, the Extender for Db2 populates the appropriate fields in the SQL Data Area (SQLDA) to contain the data, and the SQL Common Area (SQLCA) to contain the error messages and return codes for communication with the application program.

For data returning from the local Db2 subsystem:

- All retrieved data is returned to the appropriate data area in the SQLDA, as in a typical local Db2 subsystem operation.
- All the local Db2 subsystem error messages and return codes are returned in the SQLCA, as in a typical local Db2 subsystem operation.

For data returning from the server, the Extender for Db2:

- Receives a block of data from API from any single SQL request.
- Builds an answer set descriptor in the IBM SQLDA format and returns the data back to the application program, as if the application was communicating with the local Db2 subsystem.
- Transfers and communicates all server error messages to the application via the SQLCA.
  - All server error messages and codes are mapped to equivalent local Db2 subsystem error messages and codes.
  - Server messages and codes that have no Db2 equivalent are flagged using the default error number.
  - Non-Db2 mappable server messages cause the Extender for Db2 to return the generic (default) SQLCODE.
For more information about how the Extender for Db2 returns error messages, see *Using the Extender for Db2 and Db2 SQL* on page 63.
Installing the Extender for Db2 on z/OS

This section describes how to install the Extender for Db2 in a z/OS environment.

In this chapter:

- Installation Requirements
- Pre-Installation Issues
- Installation Worksheet for the Extender for Db2
- Installation Procedure
- Installing the Extender for Db2 Without Db2

Installation Requirements

This section lists the hardware, software, and system requirements for the Extender for Db2 in a z/OS environment.

Hardware Requirements

To operate the Extender for Db2, the following hardware is required.

- An IBM or IBM-compatible mainframe supporting z/OS 2.1 or higher operating systems.
- A minimum of 240 Cylinders of 3390 DASD device for the qualif.HOME.LOAD data set containing the Extender for Db2 interface load modules, the IBM Db2 module entry-points, the Extender for Db2 main module, and other Extender modules.

Software Requirements

To operate the Extender for Db2, the following minimum software levels are required.

- Server for z/OS, Version 7.7 or higher
- Db2 v11 or higher
- z/OS 2.1 or higher

When using the Extender for Db2 with a Db2 application product, see the documentation of that product for any additional requirements.
Disk Requirements

The following data sets are needed for the installation of the Extender for Db2.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Cylinders of 3380 or 3390</th>
</tr>
</thead>
<tbody>
<tr>
<td>qualif.HOME.LOAD</td>
<td>240</td>
</tr>
<tr>
<td>qualif.HOME.DATA</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:** If a PDS Deployment z/OS Server is installed in the same LPAR, you can use the existing qualif.HOME.LOAD library for the Extender for Db2.

System Requirements

Before you install the Extender for Db2, you should have access to a server, with all related data adapters and communications files installed. The server is not required for the Extender for Db2 installation procedure. However, for the Extender to function as a working client, communications between the Extender and the server are essential. For more information about setting up communications, see the *Server Administration* manual.

You must install the Distributed Data Facility (DDF). Otherwise, the QMF Draw functionality (which checks the CSECT for the presence of DDF) fails. For more information on whether you have DDF installed with your Db2 subsystem, see your system administrator.

Pre-Installation Issues

The Extender for Db2 is situated between your application (such as QMF) and the local Db2 subsystem (and the server). By providing a Db2 interface to the application, all requests from the application are intercepted by the Extender for Db2 before reaching either the local Db2 subsystem, or the server. The Extender for Db2 uses either one of two standard Db2 Attachment Facilities as the Db2 interface to your application:

- Call Attach Facility
- DSN Command Processor

Before installing the Extender for Db2, you must determine which facility is used by your application and is suitable for your application environment. For more information about the advantages and disadvantages of each facility, see the *IBM Db2 Application Programming and SQL Guide*. 
Call Attach Facility

If your application uses the Db2 Call Attach Facility (CAF), there are two ways to access the CAF language interface:

- **Explicit LOAD** on page 25
- **Link-edit of DSNALI** on page 26

The advantages and limitations of using either CAF access mode are fully explained in the *IBM Db2 Application Programming and SQL Guide*. The Extender for Db2 supports both modes of CAF access to Db2. Depending upon whether your application uses CAF explicit loading or CAF link-editing of DSNALI, you can install the Extender for Db2 to support either mode of CAF access to Db2.

**Explicit LOAD**

You can use the explicit loading feature of Db2 with the Extender for Db2. For the Extender, simply allocate the `qualif.HOMEEEXT.LOAD` library before the standard Db2 load libraries.

The modules in `qualif.HOMEEEXT.LOAD` must be invoked before the modules in the standard Db2 load library, `DSNxxx.DSNLOAD` because the library `qualif.HOMEEXT.LOAD` contains the entry points DSNALI, DSNHLI, and DSNTIAR, which have similar names to the standard Db2 entry points. If the library `qualif.HOMEEXT.LOAD` is placed before the standard Db2 load library in your JCL or CLIST, the Extender for Db2 module is invoked before the identical modules in the Db2 standard library.

Your application product allocations (such as those for QMF) are usually placed in sequence order between the Extender for Db2 libraries and the standard Db2 load library. For example, QMF requires ISPF load libraries. These are allocated between the Extender for Db2 libraries and the Db2 load library. For more information about the specific Extender for Db2 installation steps, see *Installation Procedure* on page 28.

To allocate the Extender for Db2 libraries for run-time, place the library `qualif.HOMEEEXT.LOAD` the application-specific load libraries, and the standard Db2 load libraries, in that order (referred to as the server concatenation), in your load library search path. Most installations offer the following choices:

- Create a new logon procedure and allocate the server concatenation to STEPLIB.
- For applications invoked under ISPF, allocate the server concatenation to ISPLLIB (if you are not permitted to allocate to STEPLIB).
- Allocate the server concatenation to any other ddname for which the application expects to find the standard Db2 load library allocation.
**Note:** If the chosen load library file allocation contains APF authorized data sets, the Extender for Db2 load data sets must also be APF authorized. If the Extender for Db2 load data sets are not APF authorized and are integrated with other APF authorized data sets under a ddname such as STEPLIB, the APF authorization of all the data sets is invalidated.

**Link-edit of DSNALI**

If your application link-edits DSNALI using the CAF interface to Db2, you must instead link-edit the Extender for Db2’s DSNALI interceptor stub (located in `qualif.HOMEEXT.LOAD`) to your application. For more information about the specific Extender for Db2 installation steps, see *Installation Procedure on page 28*.

**DSN Command Processor Facility**

The Extender for Db2 supports applications that communicate to Db2 via the DSN Command Processor Facility. As with the support for CAF link-edit with DSNALI, you can enable the Extender for Db2 to use the DSN Command Processor for your application. To do so, simply perform a link-editing step of the Extender for Db2’s DSNELI with your application program. For more information about the specific Extender for Db2 installation steps, see *Installation Procedure on page 28*.

**Installation Worksheet for the Extender for Db2**

The following is an overview of the steps that are performed to install the Extender for Db2, with variations based on specific user and system requirements. The worksheet that follows the overview will help you determine which steps are required for your installation.

**Overview**

1. **Step 1. Unload Two Data Sets From the Tape** on page 28
2. **Step 2. Copy Extender Entry Points** on page 29
3. **Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional)** on page 30
4. **Step 4. Configure the Parameter File EDAPARMS (Optional)** on page 31
5. **Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional)** on page 31
6. **Step 6. Prepare Run-time Allocation Streams** on page 33

**Note:** For more information on the software and hardware requirements that must be met before proceeding with installation, see *Installation Requirements on page 23*. 

---

26 Information Builders
Worksheet

Answer the questions on the following Worksheet, which is intended to help you to differentiate between installation types based on specific user and system requirements. Proceed as instructed, following the detailed information in the referenced steps, as appropriate for your installation.

The Extender for Db2 is shipped as part of the z/OS PDS Deployment Server.

**Are you planning to install the PDS Deployment Server in the same z/OS LPAR as the Extender for Db2?**

- If Yes:
  - Install the server first, then skip **Step 1. Unload Two Data Sets From the Tape** on page 28 and proceed to **Step 2. Copy Extender Entry Points** on page 29. (Not that Step 1 is not necessary because `qualif.HOME.LOAD` and `qualif.HOME.DATA` are unloaded during the PDS Deployment installation.)
  - Use the PDS Deployment EDAENV dataset, located at `qualif.PDS.server_type.DATA` (rather than the simplified one provided in the sample job streams included in this document).

- If No, run **Step 1. Unload Two Data Sets From the Tape** on page 28, followed by **Step 2. Copy Extender Entry Points** on page 29.

**Will Db2 Extender talk directly to the local Db2 subsystem (without using a server)?**

- If Yes, run **Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional)** on page 30.
  - If No, skip Step 3.

**Are you installing Db2 Extender without a local Db2 subsystem?**

- If Yes, run **Step 4. Configure the Parameter File EDAPARMS (Optional)** on page 31 to define a default server name.
  - If No, Step 4 is optional.

**Does your application require explicit link-edit to IBM’s Db2 interface modules (DSNALI, DSNELI or DSNTIAR) or does it explicitly load the interface modules?**

- If explicit link-edit, run **Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional)** on page 31.
  - If explicit load, skip Step 5.
In which mode does your application run?

**Interactive:** Are you running QMF?

- If Yes, prepare your run-time CLIST from the sample provided in *Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI* on page 33.

- If No, are you running Call Attach Facility (DSNALI) or DSN Command Processor (DSNELI)?
  
  - If DSNALI, prepare your run-time CLIST by converting the sample JCL in *Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI* on page 38 into a CLIST.

  - If DSNELI, prepare your run-time CLIST by converting the sample JCL in *Step 6.3. DSN Command Processor (TSO Attach)* on page 40 into a CLIST.

**BATCH:** Are you running QMF?

- If Yes, prepare your run-time JCL from the sample provided in *Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI* on page 33.

- If No, are you running Call Attach Facility (DSNALI) or DSN Command Processor (DSNELI)?
  
  - If DSNALI, prepare your run-time JCL from the sample provided in *Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI* on page 38.

  - If DSNELI, prepare your run-time JCL from the sample provided in *Step 6.3. DSN Command Processor (TSO Attach)* on page 40.

---

**Installation Procedure**

Refer to your answers on the installation worksheet, then proceed through appropriate flow of steps for your installation.

**Step 1. Unload Two Data Sets From the Tape**

Run an IEBCOPY or allocate and initialize your `qualif.HOME.DATA` and `qualif.HOME.LOAD` datasets.

`qualif.HOME.DATA` contains JCL procedures needed for the Extender for Db2 installation.

The JCL is
where:

**workunit**

Is the unit for the work data set.

**qualif**

Is the high-level qualifier for HOME.DATA.

**UNIT**

Specifies the unit type of the tape drive being used. CART is the default value, but other common names include 3480, TAPE, 3420, 3490.

**tapvol**

Is the volser label of the installation tape.

After this job has run, **qualif**.HOME.DATA is allocated, cataloged, and populated with the procedures and jobs needed to install the Extender for Db2.

**Step 2. Copy Extender Entry Points**

This step copies and renames Db2 Extender entry points creating a user library named **qualif**.HOMEEXT.LOAD. This user library must be concatenated ahead of the standard Db2 load library when running your application.

Edit **qualif**.HOME.DATA(EXTINST2), replacing **qualif** for the appropriate high level qualifier. Add a job card and submit the job.
Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional)

This step links the main Extender for Db2 module with Db2 entry points DSNALI, DSNHLI, DSNHLI2, and DSNTIAR. This is required to enable calls by the Extender for Db2 module to Db2 only for those applications that communicate with the local Db2 subsystem. To perform this step, modify and run `qualif.HOME.DATA(EXTLINK)`.

**Note:** If your application does not need to access the local Db2 subsystem, see Installing the Extender for Db2 Without Db2 on page 43.

A sample of `qualif.HOME.DATA(EXTLINK)` follows:
Step 4. Configure the Parameter File EDAPARMS (Optional)

Configuring the EDAPARMS parameter file is optional because the Extender for Db2 runs without it. However, you can use the EDAPARMS parameter file to set certain parameters—such as default error numbers, default servers, and continental decimal notation (CDN)—and the destination of partially-qualified names.

If your application loads the Db2 entry points dynamically, this step is not required.

To set up the EDAPARMS parameter file, create a data set named qualif.EDAPARMS. For more information about configuring an EDAPARMS file, see Configuring the EDAPARMS File on page 45.

Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional)
Your Db2 application may require a static link to the standard Db2 entry points, such as DSNALI and DSNTIAR. If you must link-edit your application with the Extender for Db2’s interceptors (DSNALI, DSNELI, and DSNTIAR) instead of Db2 entry points, run the JCL in Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your Application (Optional) on page 32.

**Note:** If you are using the Call Attach Facility with explicit load of DSNALI, skip this step and proceed to Step 6. Prepare Run-time Allocation Streams on page 33.

**Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your Application (Optional)**

After linking the Extender for Db2 modules to your application program, edit the following sample JCL member qualif.HOME.DATA(EXTLNKAP). This job stream replaces the calls to the IBM modules with calls to the Extender for Db2 modules.

- If your application uses DSNALI as the entry point to interface to Db2, substitute DSNALI for the first INCLUDE statement.
- If your application uses DSNELI as the entry point to interface to Db2, substitute DSNELI for the first INCLUDE statement.

A sample JCL member *qualif*.HOME.DATA(EXTLNKAP) follows:

```plaintext
//*****************************************************************************
//* Purpose: Link the user program with DB2 Extender
//* Substitutions:-Change "qualif" into the high level qualifier
//* for your DB2 Extender datasets.
//* -Change "userpgm" to the name of your program.
//* -Change "userentry" to the entry point of your code.
//* If your application uses DSNELI as the entry point to interface to
//* DB2, substitute DSNELI for the first INCLUDE statement.
*****************************************************************************
//LKED     EXEC PGM=IEWL,PARM=©LIST,MAP,XREF,LET©
//SYSLMOD  DD   DISP=SHR,DSN=
//EDAEXT   DD   DISP=SHR,DSN=
//SYSUT1   DD   UNIT=SYSDA,SPACE=(800,(150,50))
//SYSPRINT DD   SYSOUT=*  
INCLUDE EDAEXT(DSNALI)
INCLUDE EDAEXT(DSNTIAR)
INCLUDE SYSLMOD(userpgm) <- name of program
MODE AMODE(31),RMODE(ANY)
ENTRY userentry <- program entry point
NAME userpgm(R) <- name of program
/*
```

**Installation Procedure**
where:

`userhlq`

Is the user high-level qualifier.

`qualif`

Is the high-level qualifier for your data sets.

`userpgm`

Is the name of your program.

`userentry`

Is the entry point of your code.

### Step 6. Prepare Run-time Allocation Streams

If using the Call Attach Facility (CAF) with Explicit Load of DSNALI, follow the instructions in [Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI](#) on page 33 to prepare your run-time allocation.

If using the Call Attach Facility (CAF) with Link-edit of DSNALI, follow the instructions in [Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI](#) on page 38 to prepare your run-time allocation.

If using the DSN Command Processor (TSO Attach), follow the instructions in [Step 6.3. DSN Command Processor (TSO Attach)](#) on page 40 to prepare your run-time allocation.

### Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI

A sample CLIST and JCL are shown below for CAF explicit load of the DSNALI. To enable your application to invoke the Extender for Db2, you must tailor your logon environment or batch address space to the following allocation streams.

In addition, you must have a correctly configured communications configuration file (CLNTCS3).

In the following examples, `qualif` is the high level qualifier for your data sets. Other variables shown in the examples are site-dependent.

#### Sample CLIST

The following is an example of the CLIST necessary to deploy your client using a communicating server with CAF explicit load.

A sample `qualif.HOME.DATA(EXTCQMF)` follows:
/*-------------------------  REXX  -------------------------------*
* Purpose: Sample CLIST to run QMF with the Extender             *
*                                                                *
* Pass the following parameters at invocation:                   *
*      qualif   High level qualifier for DB2 Extender datasets.  *
*      db2hlq   High level qualifier for DB2 Libraries.          *
*      qmfhlq   High level qualifier for QMF Libraries.          *
*      dbss     DB2 Subsymtem name.                             *
*----------------------------------------------------------------*/
parse upper arg qualif db2hlq qmfhlq dbss
"ALLOC FI(DSQLLIB) DA('qualif'.HOMEEXT.LOAD'," ,
   '"qualif'.HOME.LOAD'," ,
   '"ISP.SISPLOAD'," ,
   '"qmfhlq'.SDSQLOAD'," ,
   '"db2hlq'.SDSNEXIT'," ,
   '"db2hlq'.SDSNLOAD') SHR REUSE"
"ALLOC FI(ADMCDATA) DA('GDDM.SADMCDA') SHR REUSE"
"ALLOC FI(ADMDEFS)  DA('GDDM.ADMDefs') SHR REUSE"
"ALLOC FI(ADMGDF)   DA('GDDM.SADMGDF') SHR REUSE"
"ALLOC FI(ADMSYMBL) DA('GDDM.SADSYM') SHR REUSE"
"ALLOC FI(ADMCFORM) DA('"qmfhlq".SDSQCHRT') SHR REUSE"
"ALLOC FI(ADMGGMAP) DA('"qmfhlq".SDSQMAPE') SHR REUSE"
"ALLOC FI(DSQPNLE) DA('"qmfhlq".DSQPNLE') SHR REUSE"

"ALLOC FI(DSQEDIT) NEW UNIT(SYSALLDA) CYL SPACE(1 1)
   DSORG(PS) RECFM(F B A) LRECL(79) BLKSIZE(4029)"

"ALLOC FI(DSQDEBUG) DA(*) SHR REUSE"
"ALLOC FI(DSQPRINT) DA(*) SHR REUSE"
"ALLOC F(EDADPDS) DUMMY SHR"

      /* copy EDAENV contents from sample EXTJQMF jcl */
"ALLOC F(EDAENV) DA('"qualif".EDAENV') SHR REUSE"

      /* copy EDACS3 contents from sample EXTJQMF jcl */
"ALLOC F(EDACS3) DA('"qualif".EDACS3') SHR REUSE"

      /* "ALLOC F(EDAPARMS) DA('"qualif".EDAPARMS') SHR REUSE" */
      /* copy IBITRACE contents from sample EXTJQMF jcl */
"ALLOC F(IBITRACE) DA('"qualif".IBITRACE') SHR REUSE"
"ALLOC F(FSTRACE) SYSOUT(X) RECFM(F) LRECL(132) BLKSIZE(132)"

"ALTLIB ACT APPL(CLIST) DA('"qmfhlq".SDSQCLTE')"
"ALTLIB ACT APPL(EXEC) DA('"qmfhlq".SDSQEXCE')"
address ispexec "LIBDEF ISPLLIB LIBRARY ID(DSQLLIB) STACK"
address ispexec "LIBDEF ISPMLIB DATASET ID('"qmfhlq".SDSQMLBE') STACK"
address ispexec "LIBDEF ISPLLIB DATASET ID('"qmfhlq".SDSQPLBE') STACK"
address ispexec "LIBDEF ISPLLIB DATASET ID('"qmfhlq".SDSQSLBE') STACK"

Sample JCL

The following is an example, supplied at `qualif.HOME.DATA(EXTJQMF)`, of the JCL necessary to deploy your client using a communicating server with CAF explicit load of the DSNALI.
Job Card Goes Here

Purpose: Sample JCL to run a QMF procedure in Batch.

Substitutions:
- qualif High level qualifier for DB2 Extender datasets.
- db2hlq High level qualifier for DB2 Libraries.
- qmfhlq High level qualifier for QMF Libraries.
- hostn Server's Host name or Server's IP address.
- portn TCP/IP Port number server is listening on.
- userid Owner of QMF procedure to be executed.
- qmfprocs QMF Procedure name.
- dbss DB2 Subsystem name.

SET DB2REL=db2hlq
SET QMFREL=qmfhlq

QMFBAT EXEC PGM=IKJEFT01,DYNAMNBR=30,TIME=1440,REGION=4096K
STEPLIB DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
STEPLIB DD DISP=SHR,DSN=qualif.HOME.LOAD
STEPLIB DD DISP=SHR,DSN=&QMFREL..SDSQLOAD
STEPLIB DD DISP=SHR,DSN=&DB2REL..SDSNEXIT
STEPLIB DD DISP=SHR,DSN=&DB2REL..SDSNLOAD
STEPLIB DD DISP=SHR,DSN=ISP.SISPLOAD

Extender Client Configuration File

NAME = Client Odin File
NODE = EDASERVE
BEGIN
  PROTOCOL = TCP
  CLASS = CLIENT
  HOST = hostn ;Server's Host name or IP address
  PORT = portn ;Port # server is listening on
  TRACE = 31
END
*/
Extender Environment

EDAENV DD *

EDACONF=/PDS

FSTRACE=DD:FSTRACE

EDADPDS DD DUMMY

Extender EDAPARMS File (Optional)

Extender Traces are enabled in EDAPARMS DD and output
goes to DD QXTRACE (dynamically allocated)

EDAPARMS DD DISP=SHR, DSN=qualif.EDAPARMS

API Tracing (trace output goes to DD FSTRACE)

IBITRACE DD *

SET TRACEON=ALL

FSTRACE DD SYSOUT=*, DCB=(LRECL=132, RECFM=FB, BLKSIZE=132)

Client Application Allocations

------------ TSO Datasets (Required for QMF) ---------------

SYSPROC DD DSN=&QMFREL..SDSQCLTE, DISP=SHR

SYSEXEC DD DSN=&QMFREL..SDSQEXCE, DISP=SHR

SYSTSPRT DD SYSOUT=* 

------------ ISPF DATASETS (REQUIRED FOR QMF) --------------

ISPPLIB DD DSN=&QMFREL..SDSQPLBE, DISP=SHR

ISPMLIB DD DSN=&QMFREL..SDSQMLBE, DISP=SHR

ISPSLIB DD DSN=&QMFREL..SDSQSLBE, DISP=SHR

ISPTLIB DD DSN=ISP.SISPSENU, DISP=SHR

ISPPROF DD UNIT=SYSDA, SPACE=(TRK,(9,1,4)),

DCB=(LRECL=80, BLKSIZE=3120, RECFM=FB, DSORG=PO)

------------ QMF Datasets -------------------------------

ADMGGMAP DD DISP=SHR, DSN=&QMFREL..SDSQMAPE

ADMCFORM DD DISP=SHR, DSN=&QMFREL..SDSQCHRT

ADMDEFS DD DISP=SHR, DSN=CSDDBS.QMF.ADMDEFS

------------ Datasets used by QMF -------------------------

DSQPRINT DD SYSOUT=* 

DSQDEBUG DD SYSOUT=* 

DSQUDEBUG DD SYSOUT=A, DCB=(RECFM=VBA, LRECL=125, BLKSIZE=1632)

DSQEDIT DD UNIT=SYSDA, SPACE=(TRK,(10,1,5))
Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI

Use this option if your application program requires a link-edit of the standard IBM Db2 Interface modules DSNALI and/or DSNTIAR (instead of the execution of a load macro) to call its routines.

The Extender for Db2 interface modules have the same aliases as the standard IBM modules DSNALI or DSNTIAR. If the IBM modules have been linked into your application program, then these modules must be substituted by the Extender for Db2 interface modules DSNALI or DSNTIAR residing in the `qualif`.HOMEEXT.LOAD.

- If your application accesses both a server and the local Db2 subsystem, your execution job stream must allocate the load libraries in sequence order ahead of the Db2 standard load libraries. The allocation is done in STEPLIB.

**Step 6.2 Example 1:**

```plaintext
//RUNSTEP EXEC PGM=userprogram
//STEPLIB DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//        DD DISP=SHR,DSN=qualif.HOME.LOAD
//        DD DISP=SHR,DSN=DSN810.SDSNEXIT
//        DD DISP=SHR,DSN=DSN810.SDSNLOAD
```

- If your application does not access the local Db2 subsytem, then you only need the Db2 Extender load libraries in STEPLIB.

**Step 6.2 Example 2:**

```plaintext
//RUNSTEP EXEC PGM=userprogram//STEPLIB DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//        DD DISP=SHR,DSN=qualif.HOME.LOAD
```

where:

- `userpgm`
  
  Is the name of your program.

- `qualif`
  
  Is the high-level qualifier for your data sets.
The following is a full Job stream Job stream that runs a program with Db2 Extender using the Link-Edit facility of DSNALI (CAF); it is provided at qualif.HOME.DATA(EXTCBCAF). (Notice that it uses the case described Example 1.)

```plaintext
/*/        Job Card Goes Here
/*/        */* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*/Purpose:Sample JCL to run CAF (DSNALI) program with DB2 Extender */
/*/Substitutions:                                                   */
/*/qualif    High level qualifier for DB2 Extender datasets.        */
/*/db2hlq    High level qualifier for DB2 Libraries.               */
/*/hostn     Server's Host name or Server's IP address.            */
/*/portn     TCP/IP Port number server is listening on.            */
/*/userlib   Dataset where XTDCOB program resides.                 */
/*/XTDCOB    replace with the name of your program                 */
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***/
//         SET   DB2REL=db2hlq
//CAFRUN   EXEC  PGM=XTDCOB
//STEPLIB  DD    DISP=SHR,DSN=userlib.LOAD
//         DD    DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//         DD    DISP=SHR,DSN=qualif.HOME.LOAD
//         DD    DISP=SHR,DSN=&DB2REL..SDSNEXIT
//         DD    DISP=SHR,DSN=&DB2REL..SDSNLOAD
//         */---------------------------------------------------------------
//                Extender Client Configuration File
//         */---------------------------------------------------------------
//EDACS3   DD    *
NAME        = Client Odin File
NODE = EDASERVE
BEGIN
  PROTOCOL  = TCP
  CLASS     = CLIENT
  HOST      = hostn ;Server's Host name or IP address
  PORT      = portn ;Port # server is listening on
  TRACE     = 31
END
/*
```
Step 6.3. DSN Command Processor (TSO Attach)

The Extender for Db2 provides a back end to your application and contains invocations similar to the Db2 DSN Command Processor invocations.

- Application requests destined to the local Db2 subsystem are first intercepted by the Extender for Db2, which then passes them through to the DSN Command Processor Facility of the local Db2 subsystem.

- Application requests destined for server databases are routed to the server.

In all cases, the application invokes the Extender using the calls and conventions of the DSN Command Processor Facility.
To enable your application directly to the Extender for Db2’s interface modules, instead of to the standard Db2 DSN Command Processor modules, you must link-edit the Extender for Db2 interface modules to your application. After you perform this link-edit, your application invokes the Extender for Db2’s DSN interface, which input mimics the standard Db2 Command Processor Facility.

The following is an example, provided at `qualif.HOME.DATA(EXTCBTSO)`, of the run-time allocation job stream for the client using a communicating client/server architecture with DSN.

```plaintext
//*        Job Card Goes Here
//*
//-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *--
//--Purpose:Sample JCL to run TSO (DSNELI) program with DB2 Extender*--
//--                                                                --*
//--*Substitutions:                                                   --*
//-- qualif    High level qualifier for DB2 Extender datasets. *--*
//-- db2hlq    High level qualifier for DB2 Libraries.             *--*
//-- dbss      DB2 Subsytem name.                                  *--*
//-- hostn     Server's Host name or Server's IP address.          *--*
//-- portn     TCP/IP Port number server is listening on.          *--*
//-- userlib   Dataset where TSOCOB program resides.               *--*
//-- TSOCOB    replace with the name of your program.              *--*
//-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *--
//         SET   DB2REL=db2hlq
//TSORUN EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
// DD DISP=SHR,DSN=qualif.HOME.LOAD
// DD DISP=SHR,DSN=&DB2REL..SDSNEXIT
// DD DISP=SHR,DSN=&DB2REL..SDSNLOAD
```
//---------------------------------------------------------------
//                Extender Client Configuration File
//---------------------------------------------------------------
//EDACS3   DD    *
NAME        = Client Odin File
NODE = EDASERVE
BEGIN
   PROTOCOL  = TCP
   CLASS     = CLIENT
   HOST      = hostn    ;Server's Host name or IP address
   PORT      = portn    ;Port # server is listening on
;   TRACE     = 31
END
/*
//---------------------------------------------------------------
//                Extender Environment
//---------------------------------------------------------------
//EDAENV   DD    *
FSTRACE=DD:FSTRACE
EDACONF=/PDS
/*
//EDADPDS  DD    DUMMY
//---------------------------------------------------------------
/*
//                Extender EDAPARMS File (Optional)
//                Extender Traces are enabled in EDAPARMS DD and output 
//                goes to DD QXTRACE (dynamically allocated)
//---------------------------------------------------------------
//EDAPARMS  DD  DISP=SHR,DSN=qualif.EDAPARMS
//---------------------------------------------------------------
/*
//IBITRACE DD    *
SET TRACEON=ALL
//FSTRACE  DD    SYSOUT=*,DCB=(LRECL=132,RECFM=FB,BLKSIZE=132)
//*--------------------------------------------------------------
//*               User Application Allocations
//*--------------------------------------------------------------
//XTDPRM    DD  *
DEBUG=N,BATCH=Y
/*
//DBGOUT  DD    SYSOUT=*  
//SYSOUT  DD    SYSOUT=*  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN  DD  *
  DSN SYSTEM(dbss)
  RUN PROGRAM (TSOCOB)   -  
    LIB('userlib.LOAD')  
END  
/*
//SYSIN   DD  *
SQL
SELECT COUNTRY,CAR,MODEL,BODYTYPE FROM EDASERVE.ANYNAME.CAR
END
SQL
SELECT LAST_NAME,FIRST_NAME FROM EDASERVE.ANYNAME.EMPLOYEE
END
EXIT
/*

2. Installing the Extender for Db2 Without Db2

You can install the Extender for Db2 without Db2. Db2 front-end applications can then issue
server requests and query non-Db2 and other relational and non-relational databases without
the presence of a Db2 database. An example is an installation site which has a requirement of
querying IMS and VSAM data. The Extender for Db2 enables access through a Db2 application
(such as QMF), but does not specifically require the installation to have Db2.

To install the Extender for Db2 without Db2, perform the steps described in Installation
Procedure on page 28 with the following changes:

1. Unload two data sets from the tape.
2. Copy Extender Entry points.
3. Omit Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional) on page
   30. This step enables the Extender for Db2 to communicate with Db2, which is not present.
4. Set up the parameter file EDAPARMS. Set the following parameters in this file:

   EDASERVER  = edaserver_name
   DBMS       = EDA
5. Link the Extender for Db2 interface modules with your application program (optional).
Complete Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional) on page 31 only if your Db2 application is statically linked to the Db2 standard IBM modules, such as DSNALI, DSNELI, DSNTIAR. Running this step links your application to the Extender for Db2 interceptor modules (also called DSNALI, DSNELI, DSNTIAR, etc.) instead of Db2’s.

☐ Tailor and run the JCL in Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your Application (Optional) on page 32.

6. Customize the sample CLIST/JCLs provided for the user program.

7. Run your application.
Chapter 3

Configuring the EDAPARMS File

This section describes how to set up the EDAPARMS configuration file.

In this chapter:

- Overview of the EDAPARMS File
- Creating the EDAPARMS File
- Using the EDAPARMS File

Overview of the EDAPARMS File

The EDAPARMS file is a sequential file that configures the Extender for Db2 for different products and can also contain installation-specific parameters. The file is created using the system editor and cannot be altered at run-time.

With the EDAPARMS file, you can:

- Define product work tables or partially-qualified table names. Table names appearing in the EDAPARMS file direct the Extender for Db2 to dispatch SQL requests to the local database subsystem.

- Define a generic system-wide error message for a server return error that is other than the default.
  
  "-901 System Error Has Occurred"

- Define a default server for partially-qualified table names. If no server was specified in an SQL CONNECT TO request, the default server name is used. If an SQL CONNECT RESET was issued, then this default server is referenced.

- Enable traces for Db2 Extender.

Use of the EDAPARMS file is optional. It can be omitted from an installation without adversely affecting the operation of the Extender for Db2.
Creating the EDAPARMS File

To set up the EDAPARMS parameter file, create a data set named qualif.EDAPARMS

* EDAPARMS file
   ERRNUM = errornum
   EDASERVER = defserver
   DBMS = database_type
   DECIMAL = decimal_point_type
   DATE = date_format
   ENABLE = trace_level
   WORKTABLE = worktablename
   .
   .
   WORKTABLE = worktablename

where:

*

Denotes comments within the EDAPARMS file. An asterisk (*) in column one identifies the line as a comment. More than one comment is possible within the EDAPARMS file.

ERRNUM

Represents an Extender for Db2 error number. The value supplied must be an integer within the range -2147483647 to 2147483648. (-901) is the default value; the Extender for Db2 uses it for server error codes that cannot be mapped to Db2 error codes. It is recommended that you assign a negative three- or four-digit number for simplicity. Avoid using a valid Db2 error code.

If more than one ERRNUM is defined, the last definition is used.

EDASERVER

Is the default server for unresolved partially-qualified tables. Verify that this server is defined in both client and server communications configuration files. This field can be up to eight bytes in length.

If more than one EDASERVER is defined, the last definition is used.

If the EDASERVER keyword references a server not defined in the communications configuration file, an error message is produced.

DBMS

There are three valid values for this parameter.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOTH</td>
<td>Enables the Extender to access both the local database subsystem and databases residing on the server. BOTH is the default value.</td>
</tr>
<tr>
<td>NATIVE</td>
<td>Enables the Extender to access only the local database subsystem and prohibits access to databases residing on the server.</td>
</tr>
<tr>
<td>EDA</td>
<td>Enables the Extender to access only databases residing on the server and prohibits access to the local database subsystem.</td>
</tr>
</tbody>
</table>

If more than one DBMS is defined, the last definition is used.

**DECIMAL**

Enables you to specify whether you want a period or a comma for the decimal point in a numeric value. PERIOD is the default value. If you specify the value COMMA here, it allows either a period or a comma to be used as the decimal point.

If more than one DECIMAL is defined, the last definition is used.

**DATE**

Enables you to specify the output format of a date field. There are four valid values for this parameter.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO</td>
<td>The output format is YYYY-MM-DD. For example, the output could be: 1992-05-21. ISO is the default value.</td>
</tr>
<tr>
<td>USA</td>
<td>The output format is MM/DD/YYYY. For example, the output could be: 05/21/1992.</td>
</tr>
<tr>
<td>EUR</td>
<td>The output format is DD.MM.YYYY. For example, the output could be: 21.05.1992.</td>
</tr>
<tr>
<td>JIS</td>
<td>The output format is YYYY-MM-DD. For example, the output could be: 1992-05-21.</td>
</tr>
</tbody>
</table>

If more than one DATE is defined, the last definition is used.
ENABLE

Enables several levels of tracing for Db2 Extender. API traces are written to DDNAME FSTRACE and all other traces are written to DDNAME QXTRACE (dynamically allocated when needed). Possible trace values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGES</td>
<td>Corresponds to the EDALOG file used in earlier releases. Users can record all warning and error messages originating from the server or the Extender for Db2. EDALOG is useful if a display or output of warnings and messages is desired. This trace is especially useful for Db2 application products that do not use the DSNTIAR message formatting facility.</td>
</tr>
<tr>
<td>SQLTRACE</td>
<td>Corresponds to the QXSQLTRC file used in earlier releases. Displays the SQL statement types (NATIVE or EDA). In addition, users can see how a statement string is physically prepared for dispatching.</td>
</tr>
<tr>
<td>RDITRACE</td>
<td>This is the main component trace for Db2 Extender. It shows the inputs received, the processing of SQLDA, and the proper handling of every query and its results.</td>
</tr>
<tr>
<td>PRSTRACE</td>
<td>SQL Parser output. Displays command type (for example, SELECT, INSERT, UPDATE, etc.), number of tables, and table names.</td>
</tr>
<tr>
<td>APITRACEx</td>
<td>Where x is from 1 to 9 API trace levels. It can be enabled separately for levels 1 to 9.</td>
</tr>
</tbody>
</table>

WORKTABLE

Is the name of a product worktable (for example, Q.PROFILE). The Extender directs all requests referencing these worktables to Db2. More than one WORKTABLE keyword definition can exist in the EDAPARMS file. For more information on naming conventions and worktables, see Using the Extender for Db2 and SQL on page 55.
Using the EDAPARMS File

The following guidelines are helpful when using the EDAPARMS file.

- Assign an ERRNUM if an application-specific error code is desired for all server errors that cannot be mapped to the local Db2 subsystem error code set.

- Assign an EDASERVER for the default destination for requests containing partially-qualified table names (two-part names) to be a specific server. If EDASERVER is omitted, the local Db2 subsystem is the default destination for requests containing partially-qualified table names. When an EDASERVER is assigned, it can be overridden by using an explicit SQL CONNECT TO verb to connect to the desired server for partially-qualified table names. The EDASERVER keyword is for default, unresolved destinations to assign to requests containing partially-qualified table names. For more information, see Connecting to Multiple Servers on page 115.

- Assign WORKTABLEs if you have already assigned an EDASERVER, but want to send requests containing specific partially-qualified table names to the local Db2 subsystem. WORKTABLEs can be product-specific partially-qualified table names that reside in the local Db2 subsystem from a product installation. Essentially, the WORKTABLE keyword overrides the effects of an EDASERVER assignment for partially-qualified table names for product-specific tables. You can assign a WORKTABLE without an EDASERVER assignment present. However, this is not necessary, as the default destination for partially-qualified tables is the local Db2 subsystem.

- To allocate the EDAPARMS file when using the Extender for Db2, add the following statement to the client CLIST

```cli
ALLOC F(EDAPARMS) DA('qualif.EDAPARMS') SHR
```

or add the following statement to the client JCL:

```jcl
//EDAPARMS DD DSN=qualif.EDAPARMS,DISP=SHR
```
This section describes how to use security and tracing with the Extender for Db2 in the z/OS environment.

In this chapter:
- Security
- Tracing

Security

Security is implemented at the server level. The Extender for Db2 provides an explicit means to access the database engine by supporting an EDA IMMEDIATE command. This command allows an SQL user to:

- Issue directives to the server identifying the user and password.
- Validate the user to a server security subsystem.
- Verify entry to specific adapters.

Using the EDA IMMEDIATE Command

The syntax of the EDA IMMEDIATE command is

```
EDA [server] command
```

where:

- `server`
  - Is optional. If included, it is the name of a valid server defined in the client communications configuration file. A value for this parameter must be supplied to override the default server name defined in the EDAPARMS file or if the SQL CONNECT command is not used.

- `command`
  - Is a valid EDA IMMEDIATE command as described below.

The proper usage of this command is to assign it to a host variable string, and issue an EXECUTE IMMEDIATE macro on the host variable string, as illustrated in the following example.
**Supported Types of Security**

There are three types of Client-supported security. To enable each type of security, use the EDA IMMEDIATE command. For more information, see *Using the EDA IMMEDIATE Command* on page 51.

- **General server user ID/password.** Used to provide a user ID and password to be validated by the server. When using this type of security, the following command must be executed before any other SQL command is sent to the server.

  string="EDA edaserver SET EDAUSER=id EDAPASS=password"
  EXEC SQL
  EXECUTE IMMEDIATE :string
  END-EXEC

  This command does not initiate communication with the server.

- **Server DBA password.** Used to set the DBA password as the general server password. This option provides file-level security. When using this type of security, issue the following command.

  string = "EDA edaserver SQL SET PASS = dbapassword"
  EXEC SQL
  EXECUTE IMMEDIATE :string
  END-EXEC

  For detailed information, see *Providing Data Source Security: DBA* in the *Describing Data With WebFOCUS Language* manual.

- **Adapter-specific password security.** Used with adapters that need specific passwords. For example, the format for setting a logon string directed at an adapter interfacing with a Teradata database is:

  string="EDA edaserver SQL SQLDBC SET DBCLOGON tdpid/userid,pwd;"
  EXEC SQL
  EXECUTE IMMEDIATE string
  END-EXEC

  In the previous string the value of command is:

  SQL SQLDBC SET DBCLOGON tdpid/userid,pwd;
For security administration, command can be set to any literal as required by the specific adapter.

**Client Security**

If your server security is enabled, the client or the Extender for Db2 must identify itself with a user ID and password, in order to access the server. For more information, see *Supported Types of Security* on page 52. For example, in z/OS/QMF, the user would type in query mode

```
EDA edaserver SET EDAUSER=userid EDAPASS=password
```

where:

- **edaserver**
  - Is a valid aerver name, as identified in the client configuration file as partner.lu.name.

- **userid**
  - Is a valid user ID known and acceptable to the server.

- **password**
  - Is a valid password known and acceptable to the server.

In QMF, the user does not need to specify EXECUTE IMMEDIATE since this is done by QMF internally. For other Db2 applications, check to confirm that EXECUTE IMMEDIATE is implicitly performed; otherwise, you may need to specify it explicitly. For more information, see *Supported Types of Security* on page 52.

**Tracing**

You can enable two types of traces:

- **Extender traces.** To enable these traces, see the instructions under the EDAPARMS file (as described in *Configuring the EDAPARMS File* on page 45).

- **API traces.** To enable these traces, in addition to enabling traces under EDAPARMS you must allocate the following DDNAMES:

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSTRACE</td>
<td>Allocate FSTRACE with the following DCB parameters:</td>
</tr>
<tr>
<td></td>
<td><code>RECFM=FB LRECL=132 BLKSIZE=13200</code></td>
</tr>
<tr>
<td>DDNAME</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IBITRACE</td>
<td>IBITRACE is a one line file containing the following command:</td>
</tr>
<tr>
<td></td>
<td>SET TRACEON=ALL</td>
</tr>
<tr>
<td>EDAENV</td>
<td>Make sure the following line is coded:</td>
</tr>
<tr>
<td></td>
<td>FSTRACE=DD:FSTRACE</td>
</tr>
</tbody>
</table>

**Note:** You can allocate these three files permanently to the jcl streams since traces will only be produced if they are enabled from EDAPARMS file.

### QXUDUMP Data Set File Allocation

With QXUDUMP, users can produce a dump data set containing the results from an Extender for Db2 abend. Users requiring assistance to interpret the contents of this dump should contact Customer Support Services.

The following DCB characteristics for the CLIST or JCL allocation for QXUDUMP for a 3380 device are:

- **RECFM=VBA**
- **LRECL=125**
- **BLKSIZE=882**

The CLIST allocation is:

```
ALLOC F(QXUDUMP) DA('qualif.QXUDUMP') MOD -
RECFM(VBA) LRECL(125) BLKSIZE(882)
```

The corresponding JCL allocation for QXUDUMP is:

```
//QXUDUMP    DD   DSN=qualif.QXUXTDUMP,DISP=SHR
            DCB=(RECFM=VBA,LRECL=125,BLKSIZE=882)
```

### Error Message Formatting Facility (DSNTIAR)

DSNTIAR is a message formatting facility feature of Db2. A typical Db2 application invokes DSNTIAR to convert an SQLCODE to a text message only if DSNTIAR is statically link-edited with the application program.

The Extender for Db2 also supports a DSNTIAR entry point. This feature enables conversion of both the SQLCODEs from Db2 and the return codes from the server to meaningful text messages.
This section describes Translation Services and how the Extender for Db2 compares to ANSI SQL.

In this chapter:
- Table Naming Conventions
- SQL Translation

Table Naming Conventions

Users must follow table naming conventions when accessing tables. SQL requests must be structured to contain either:

- All tables accessible under one server.
- All tables accessible under the local or default Db2 subsystem.

Mixed destination requests containing both local Db2 subsystem tables and database tables accessible to the server are not permitted.

The Extender for Db2 directs each SQL SELECT statement issued against a table based on the set of rules outlined in this section. There are three types of tables known to the user.

- Fully-Qualified Tables on page 55
- Partially-Qualified Tables on page 56
- Product Work Tables on page 57

Fully-Qualified Tables

Fully-qualified table names consist of three-part names and contain an explicit location. The syntax for a fully-qualified table is

`location.creator.tablename`
where:

**location**

Is the location of the data. If location is one of the current servers in the Client communications configuration file, then the location is a valid server location. This is an 8-byte (character) field.

**creator**

Is the creator of the table. This is an 8-byte (character) field. Currently, it is ignored for tables residing in the server.

**tablename**

Is the name of the table. For the server, it must be a Master File name or a system-defined catalog name.

For example,

**EDASERVE.JAMES.EMPLOYEE**

indicates that the table EMPLOYEE was created by JAMES and can be accessed through a server called EDASERVE. The name of the server is determined at installation. The server’s communications configuration file contains the name of the server and is established at installation.

**Partially-Qualified Tables**

Partially-qualified table names have one- or two-part names and do not contain an explicit location. For partially-qualified table names, the Extender for Db2 parses the SQL CONNECT TO server command to determine whether to route the request directly to Db2, or to the server. Partially-qualified table names can be listed in the EDAPARMS configuration file under the WORKTABLE keyword.

The syntax for a partially-qualified table is

**creator.tablename**

where:

**creator**

Is the creator of the table. This is an 8-byte (character) field. It is ignored for tables residing in the server.

**tablename**

Is the name of the table. For the server, it must be a Master File name or system-defined catalog name.
For example,

WATSON.SUPPLIER

indicates that the SUPPLIER table was created by WATSON.

While parsing the SQL request, the Extender for Db2 determines if the object has a one-part or two-part name. If so, the Extender for Db2 handles the object as a partially-qualified table. Based upon the most recent SQL CONNECT TO issued, the Extender for Db2 resolves the destination of the partially-qualified table name to be the current server.

- If an SQL CONNECT TO is issued to a valid server, the request is sent to the server. Otherwise, it is sent to Db2.

- If no SQL CONNECT TO was issued, the Extender for Db2 checks the EDAPARMS file to determine if a default server was declared using the EDASERVER keyword. For information about the contents of the EDAPARMS file, see Configuring the EDAPARMS File on page 45.

  - If a default server is declared in EDAPARMS, the request goes to that default server. However, if the partially-qualified table name matches a worktable specified in the EDAPARMS file, the default server is overridden and the request is sent to the local Db2 subsystem.

  - If there is no default server and no worktables are defined, the request is sent to Db2. For more information on support for the SQL CONNECT verb, see Connecting to Multiple Servers on page 115.

**Product Work Tables**

Each product that uses the Extender for Db2 can have its own work tables and profile logs, all located outside of the server. They must be listed in the WORKTABLE keyword list in the EDAPARMS configuration file. The Extender for Db2 considers all of them non-server and sends corresponding queries referencing these tables to Db2.

Table names should be 8-byte (character) fields. The rules in Partially-Qualified Tables on page 56 also apply to naming product work tables.

**SQL Translation**

This section describes how the Extender for Db2 and the SQL Translator access heterogeneous relational and non-relational databases. The topics include:

- Column Name Resolution on page 58
Column Name Resolution

When resolving column names, the SQL Translator does not accept a unique truncation of a column name as a valid name for that column. For example, if you had a table with a column named EMPID, you cannot refer to that column as EMP (assuming no other column referenced in the request began with those three letters). You must refer to the column by its full name, EMPID.

Alternate Column Names

The SQL Translator is fully ANSI compliant; therefore, the user cannot use ALIAS= facility in the Master File. The ANSI specification states that a column has only one name. To rename a column logically, the application should create a view of that table with a different column name.

Dynamically Defined Virtual Fields

It is not possible to define a virtual field dynamically in a remote procedure and use it in subsequent SQL statements against that table. Virtual fields must be defined in the Master File. For more information about Master Files, consult the server Adapter Administration manual.

Answer Set Generation

To provide completely transparent SQL access, SQL Translation Services create a Cartesian product style answer set in all cases, regardless of the nature of the underlying DBMS. A Cartesian product style answer set is in keeping with the SQL-based nature of the data access mechanism.

What Is a Cartesian Product?

A Cartesian product or set multiplication is defined as the pairing of each element of x with every element of y. This type of response is the expected result of a relational JOIN.
What Does This Mean to You?

This means that in some instances, the answer sets received are larger than expected. The Cartesian product generation only affects situations involving JOINs, either implicit or explicit.

- **Explicit join.** Defined in the SQL statement used to generate the answer set.
- **Implicit join.** Reference to any data structure made up of independent parts, such as segments in a hierarchy.

For example, there is a three-segment hierarchical database. The top segment represents departments, one child segment represents employees, and the other—the furniture used by that department. Assume that the payroll department has 20 employees and 22 desks. If you ask for all of the employees and furniture from the payroll department, the SQL user would expect to get each employee listed 22 times, once for each desk. This type of answer is the Cartesian product set answer, and is consistent with the result you would expect from an SQL-based DBMS, such as Db2.

The Cartesian product set answer can appear only under certain specific circumstances, as in the above example, with multi-path requests in a hierarchical data structure. In general, it only results in a repetition of rows.

Answer Set Generation Logic

The algorithm used to interpret the generation of answer sets is straightforward. This algorithm is the structure around which answer set generation is performed. This algorithm will be familiar to any experienced SQL user, but may be a new experience to application developers and users that are more familiar with other DBMS systems.

This algorithm does not correspond to the internal mechanism of generating answer sets, but is a convenient means of thinking about that process. The internal mechanisms are different because they have been optimized for performance in specific DBMS environments.

The (simplified) algorithm is:

1. Create the Cartesian product of every logical table referenced in the answer set. A logical table is defined as:
   - A relational table or view.
   - A flat file (VSAM, C-ISAM, etc.).
   - A segment of a hierarchical database.
   - A segment of a network database.
   - Any other data structure designated as a segment in the Master File.
2. Remove all rows from the Cartesian product that do not pass the screening criteria specified in the WHERE clause of the SQL statement.

3. Calculate any valued expressions in the SQL statement.

4. Perform the ordering and grouping specified in the SQL statement.

5. Remove any repeated values if specified with the DISTINCT operator.

6. Calculate the results of any column functions (SUM, COUNT).

7. Remove the result rows that do not correspond to screening conditions in the HAVING clause of the SQL statement.

8. Return the answer set.

**What to Look For**

Three things help to explain the Cartesian product answer set generation, particularly for users unfamiliar with SQL-based DBMSs.

- Results of aggregate functions, such as sum or count, are generated after the Cartesian product is created. This means that in the department/employee/furniture example, if the user requests a sum of the employees' salaries in every department that had enough desks for all employees, they receive the result of 22 times the sum of the salaries for the payroll department. This is because each employee is associated with each desk, and each desk with each employee, resulting in 440 (20 X 22) items in the Cartesian product instead of 20.

- The Cartesian product is generated for the referenced logical tables in the request. For a hierarchical database, this refers to the referenced subtree. If the department/employee/furniture request is changed to ask for only a sum of the salaries of the employees, with no reference to the desks, the result is the expected sum of the salaries. Since the desks were not referenced, the employees would not be repeated in the personnel department.

- A row is created only when every logical table in the join exists. If you ask for the sum of salaries where there are enough desks, you do not get a result for departments that did not own any desks. This behavior is usually referred to as an *inner join*.

**Additional Features for SQL Translation Services**

- **ANSI Level 2.** The SQL Translator is compliant with the ANSI Level 2 SQL definition.

- **Virtual Column Support.** The SQL Translator supports the definition of virtual column in the Master File, and the use of these in any capacity in which you would use a regular database column.
SQL Join Improvements. The SQL Translator handles virtually any join predicate based on an equality condition, regardless of the indexing or other characteristic of the column. This eliminates the necessity to understand any of the characteristics of the DBMS in which the data is stored.

View Creation/Deletion. You can CREATE and DROP temporary views in server databases.

SQL Translation Performance Enhancements. The SQL Translator provides improved functionality and performance. A sophisticated JOIN optimizer is included.

These and other features allow server users to develop the client/server applications quickly and easily.

SQL Translation Services Limitations

The following limitations exist in the Extender for Db2.

1. A maximum of 16 tables may be referenced in a single SQL statement.
2. A maximum of six SELECT statements may be joined by the UNION operator.
3. Correlated subqueries are not supported, except for Db2 tables.
4. The maximum number of columns that may be in the column list of a SELECT statement is 256. Your actual limit for a given query may be less, because the SQL Translation Services may reserve several of these items for its own use.
5. Date and time arithmetic are not supported.
6. The maximum number of fields in a GROUP BY or ORDER BY clause is 32. Again, the SQL Translation Services may reserve a small number of these for its own use.
7. The maximum length of an SQL statement is 4000 bytes.
8. The maximum size of a row is 32K.
This section describes enhancements and limitations of the Extender for Db2 when working with Db2 SQL.

**In this chapter:**

- General Considerations
- Discrepancies Between the Server and Db2 SQL
- Enhancements to Db2
- SAA CPI Functionality Checklist
- Extender for Db2 Cross-reference Table

**General Considerations**

Db2 applications enabled with the Extender for Db2 find differences between server and Db2 SQL. These differences involve syntax, semantics, and error message reporting. The Extender for Db2 user must be aware of these differences while preparing a Db2 application, in order to take full advantage of server capabilities and to abide by its limitations. This chapter describes the differences between server and Db2 SQL, so that Extender for Db2 users can utilize the interface effectively.

As a general rule, all SQL must follow the ANSI Level 2 standard. The server encompasses the ANSI Level 2 standard, plus some Db2 extensions. However, not all extensions are supported. These notes pertain to the following table, which lists SQL considerations.

- Tables A and B are fully-qualified server tables.
- ESRVx are specific server location names.
- Cx specifies a column name.
- Tx specifies a correlation tag.
<table>
<thead>
<tr>
<th>Description</th>
<th>Examples</th>
<th>Messages trace</th>
</tr>
</thead>
</table>
| Joins between two tables residing under different servers are not possible. | **Valid:**

`SELECT T1.C1, T2.C2 FROM ESRV1.X.A T1, ESRV1.X.B T2 WHERE T1.C1 = T2.C2;`

**Invalid:**

`SELECT T1.C1, T2.C2 FROM ESRV1.X.A T1, ESRV2.X.B T2 WHERE T1.C1 = T2.C2;`

Cannot refer to multiple servers. |
| Joins between a local Db2 subsystem (see note) with a server-accessible table is not supported. | **Valid:**

`SELECT T1.C1 FROM ESRV.X.TABL T1, ESRV.DB2.TABL T2 WHERE T1.C1 = T2.C2;`

**Invalid:**

`SELECT T1.C1 FROM DB2.TABL T1, ESRV1.X.TABL T2 WHERE T1.C1 = T2.C2;`

None. The user receives SQLCODE=-512 from Db2. |
| GROUP BY fields must be referred to by name, not by positional value.       | **Valid:**

`SELECT C1, C2, C3 FROM A GROUP BY C1, C2, C3`

**Invalid:**

`SELECT C1, C2, C3 FROM A GROUP BY 1, 2, 3`

FOC14069 SQL syntax error. |
<table>
<thead>
<tr>
<th>Description</th>
<th>Examples</th>
<th>Messages trace</th>
</tr>
</thead>
</table>
| Correlated subqueries supported only for Db2 tables. | **Valid:**

```
SELECT DISTINCT T1.C1, T1.C2
FROM A T1, B T2
WHERE T1.C1 < '00100'
AND
   T1.C2 = T2.C2 AND
T2.C3 = 'EDA';
```

**Valid:**

```
SELECT C1,C2
FROM A
WHERE C1 < '00100' AND
C2 IN (SELECT C2
FROM B
WHERE C3 = 'EDA');
```

**Invalid:**

```
SELECT T1.C1, T1.C2
FROM A T1
WHERE T1.C1 < '00100'
AND
EXISTS (SELECT 1
FROM B
WHERE C2 = T1.C2
AND   C3 = 'EDA');
```

(valid query if both A and B are DB2 tables defined on the EDA/SERVER) | EDA14013

UNSUPPORT-ED SYNTAX: Correlated Subquery SQL CODE=-84.

Description Examples EDALOG Message.
Maximum number of columns per table is 256. (The actual limit for a given query may be less, because the SQL Translation Services may reserve several of these items for its own use).

<table>
<thead>
<tr>
<th>Description</th>
<th>Examples</th>
<th>Messages trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid:</td>
<td><code>SELECT A.name, A.name...(250 times) FROM A</code></td>
<td>EDA00005 THE NUMBER OF VERB OBJECTS EXCEEDS THE MAXIMUM, which maps to SQLCODE=-840. Too many columns in a request.</td>
</tr>
<tr>
<td>Invalid:</td>
<td><code>SELECT A.name, A.name...(750 times) FROM A</code></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

- Local Db2 subsystem refers to the Db2 subsystem normally accessed via dynamic SQL by a Db2 application independent of the server. Usually, two-part table names are used for this type of access. To access server data, three-part names are used, with the first part referencing a valid server location. By default, the Extender for Db2 enables access to both the local Db2 subsystem and the server data in separate requests using the respective naming conventions, but does not support the mixing of naming conventions where the location names are different in the same request.

- Maximum supported size of a decimal column is 31-digit decimal columns.

- If data types are equal, arithmetical operations return values in the same data type of the operands, otherwise a FLOAT(15,3) is returned. If you observe unexpected overflow indicators in your data, make sure that the USAGE specification in your server’s Master File for the table is sufficiently large for the data type being displayed.

**Data Conversion**

This section describes supported conversions of data returning from a request to data described in an SQL Data Area (SQLDA). Users must verify that field types returned by a request from a table are compatible with field types designated in the SQLDA. The application communicating with the Extender for Db2 is assumed to receive the data from the SQLDA.
The data retrieved by the adapter is converted to a server format. The Extender for Db2 then converts the data from the server format to Db2 format and returns the data in a standard SQLDA. The rules for converting data retrieved by the adapter are described in the server Adapter Administration manual. The rules for converting data in a server format to Db2 format are described in this section.

In general, format conversion between similar types (such as numeric to numeric and character to character) is preferred and, when appropriate, is performed. This is illustrated in the following table.

<table>
<thead>
<tr>
<th>Server Format</th>
<th>Db2 Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>INTEGER, SMALLINT, FLOAT, DECIMAL</td>
</tr>
<tr>
<td>Alphanumeric</td>
<td>VARCHAR, CHAR</td>
</tr>
<tr>
<td>Zoned</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>Date</td>
<td>ISO, USA, EUR, JIS</td>
</tr>
</tbody>
</table>

**Note:** Db2 normally overwrites, truncates, or does not access the data area. The Extender for Db2 operates in a similar capacity.

**Error Handling**

Error messages originating from the server are converted to an appropriate Db2 error code. This Db2 error code is then returned to the application via the SQLCA. Applications communicating with the Extender for Db2 are assumed to receive error messages and tokens via the SQLCA.

Error codes returned to a client application by the Extender for Db2 are communicated using the SQLCA. There are three sources that can generate error messages.

- Extender for Db2—internal messages from the Extender for Db2.
- Server API—status codes (for example -9, -12).
- Server—internal messages from the server (for example EDA251, EDA757).

The Extender for Db2 attempts to map all messages originating from the server to an appropriate SQLCODE and tokens, and returns this information back to the application via SQLCA’s sqlcode and sqlerrmc fields. For more information on the current mapping of the messages, see *Extender Error Messages and Codes* on page 111.
If there is no known mapping to an appropriate Db2 error code, the Extender for Db2 returns a generic system error message. The following message indicates a system error:

-901 A SYSTEM ERROR HAS OCCURRED

The Db2 code can be overwritten by the ERRNUM keyword in the EDAPARMS file. If users receive the generic error message on the screen, they may allocate an EDALOG DD card on the client address space to resolve the error message further.

Parameter Marker Support

The Extender for Db2 provides limited support of parameter markers with dynamic SQL, as described in the IBM Db2 Application Programming and SQL Guide.

Applications using parameter markers must first DECLARE the cursor, issue a CONNECT to the server, PREPARE the dynamic SQL SELECT request, and perform an OPEN cursor using an SQLDA, with the following example format:

EXEC SQL OPEN C1 USING DESCRIPTOR SQLDA1

Note: Non-SELECTs using EXECUTEs are not supported. Also, host-variables in the USING clause of the OPEN are not supported.

Discrepancies Between the Server and Db2 SQL

The server is modeled after the SQL standard as defined by ANSI Level 2 (with some Db2 extensions). Db2 users may find some discrepancies between the server and Db2 SQL. The discrepancies between the server and Db2 SQL are in the specification of requests, and in expecting certain data answer set displays and Db2-like error messages.

These discrepancies are in the following categories.

- Db2 Non-ANSI Compliant SQL Requests on page 69.
- Answer Set Displays on page 70.
- Db2 Error Codes (SQLCODEs) on page 70.

To ensure smooth usage and operation, users should be familiar with these discrepancies before using Db2 applications with the Extender for Db2.
Db2 Non-ANSI Compliant SQL Requests

Db2 non-ANSI compliant SQL requests are specific to Db2 SQL and do not conform to ANSI Level 2 SQL. The server follows the ANSI Level 2 SQL standard. The following lists unsupported SQL requests.

1. Blank spaces placed before, in the middle of, or after a table name surrounded by quotation marks are unsupported and are treated as significant by the Extender for Db2, for example:

```
SELECT * FROM "EDASERVE"."X"."TABLENAME   
SELECT * FROM "EDASERVE"."X"."   TABLENAME"  
SELECT * FROM "EDASERVE"."X"."TABLE   NAME"
```

These requests return an SQLCODE=-901 with EDALOG message (-901 is the default value for ERRNUM):

EDA14063 TABLE NAME CONTAINS ILLEGAL CHARACTER

2. Invalid expressions used with aggregate functions return an EDALOG message

EDA14007 SYNTAX ERROR AFTER...

which maps to SQLCODE=-104. Db2 produces an answer set for the following example SQL statement:

```
SELECT AVG(DISTINCT NINTPART/QAVALINV) 
FROM EDASERVE.X.VPARTINV A
```

3. VARCHAR, GRAPHIC, DECIMAL, FROM, WHERE, and other standard SQL keywords used as table names in the three-part name, return an EDALOG message

EDA14007 SYNTAX ERROR AFTER...

which maps to SQLCODE=-104. Examples of SQL statements that generate this error message are:

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM EDASERVE.X.VARCHAR</td>
<td>Db2 returns data if table X.VARCHAR exists.</td>
</tr>
<tr>
<td>SELECT VARGRAPHIC(NINTPART) FROM EDASERVE.X.VPARTINV</td>
<td>Db2 returns SQLCODE=-171.</td>
</tr>
<tr>
<td>SELECT TIMESTAMP FROM EDASERVE.X.VPARTINV</td>
<td>Db2 returns data.</td>
</tr>
</tbody>
</table>

4. Db2 does not allow the creation of a table with hyphenated column names. The server allows hyphenated column names in the Master File. However, using the Extender for Db2 to query a defined hyphenated column results in the EDALOG message
EDA1400 SQLCODE IS -206

which is equivalent to:

*Column not found in the named table*

This result is correct when querying a Db2 table via the server.

**Answer Set Displays**

Answer set display discrepancies are comprised of those resultant SQL answer set formats that differ from those normally expected by Db2 users (such as placement of null data value during sorting). Answer set display considerations are:

1. For columns displayed as a result of an arithmetical computation or an aggregate function, column names are generated. These names are SQLDEF01, SQLDEF02, etc. They appear in the SQLDA in the answer set as the column name of the computed data column.

2. When a sort is executed in ascending order, null data appears at the end of the resulting answer set. In Db2, null values are expected to appear at the top of the answer set. The opposite is true if the sort is executed in descending order.

**Db2 Error Codes (SQLCODEs)**

In some cases, Db2 error codes (SQLCODEs) generated from server applications differ from those returned by Db2 for similar SQL statements. This is because the server has a different parsing mechanism for SQL requests than Db2, and cannot duplicate some Db2-specific features. In other cases, the server returns an SQLCODE equivalent to that returned by Db2.

For more information on cross-referencing SQLCODEs with the expected Db2 codes, see *Extender for Db2 Cross-reference Table* on page 79, which contains examples of SQL statements that generate the Extender for Db2 SQLCODE. SQLCODEs are discussed here.

1. If a column in the ORDER BY list is not contained in the SELECT list, the server returns an SQLCODE=-206 for a column not found, or SQLCODE=-901 for a column which exists for a table. Db2 would return an SQLCODE=-208.

2. When a numeric or date/time column is specified in a LIKE predicate, the server returns an SQLCODE=-132

   *INVALID LIKE PREDICATE*

   whereas Db2 would return an SQLCODE=-414. An example of this type of statement is:

   ```
   SELECT * FROM EDASERVE.X.VENDPART
   WHERE NVENPART LIKE '%700%'
   ```

3. Invalid use of ">" returns SQLCODE=-104, with
EDA14007 SYNTAX ERROR AFTER...

Db2 returns SQLCODE=-115

Invalid use of '-navbar'

An example of a statement that produces such an error is:

```sql
SELECT *
FROM EDASERVE.X.VPARTINV
WHERE NINTPART > ANY QAVALINV
```

4. An exponential value exceeding limits in the SELECT clause returns

EDA202 INTERRUPT. FLOATING VALUE OVERFLOW

which is mapped to SQLCODE=-802

Exception error on arithmetic operation.

Db2 returns SQLCODE=-405

Numeric literal out of range.

An example of an SQL statement that generates this error message is:

```sql
SELECT (NINTPART - 7.3E75)
FROM EDASERVE.X.VPARTINV A
```

5. An invalid WHERE clause using LIKE "%%" returns an EDALOG message

EDA14007 SYNTAX ERROR AFTER...

which maps to SQLCODE=-104. Db2 returns SQLCODE=-312

Undefined or unusable host variable.

An example of an SQL statement that generates this error message is:

```sql
SELECT DISTINCT *
FROM EDASERVE.X.SYSCOLUM
WHERE NAME LIKE "%%"
```

Enhancements to Db2

The Extender for Db2 provides the following enhancements to Db2. Generally, the server follows the ANSI Level 2 standard. In certain situations, the following server features may be preferable:

1. Comparing a literal longer than the column definition in the Master File results in an EDALOG message
EDA0015 TEST VALUE IS LONGER THAN THE FIELD FORMAT LENGTH

which maps to an SQLCODE=-901. Db2 performs the comparison and always returns a false.

An example of an SQL statement that generates this error is:

```sql
SELECT *
FROM EDASERVE.X.VPRODUCT
WHERE NPRODUCT = 'BBDDO'
```

where the Master File for the field NPRODUCT is a 4-byte character.

2. If a column function expects a column name and does not find it, but finds an arithmetic computation instead, the server performs the computation and returns the result multiplied by the number of rows in the table. For example,

```sql
SELECT SUM(10*10)
```

The server returns 3500 as a result for a table with 35 rows (that is, 10*10*35). Db2 returns an SQLCODE=-111

Column function does not include a column name.

To justify this behavior, Db2 also returns 3500 as a result for a table with 35 rows for `SELECT SUM(10 *10 + SEATS - SEATS),` where SEATS is a defined column of the table.

3. Creator names that exceed 8 characters generate an error message in Db2. The server accepts creator names greater than 8 characters for some relational databases, other than Db2.

4. Nested aggregate functions, which are not permitted in Db2, return SQLCODE=-112

Operand of a column function is another column function or DISTINCT followed by an expression.

An example of an SQL statement that uses a nested aggregate function is:

```sql
SELECT AVG(ID*MAX(COMM))
FROM EDASERVE.Q.QSTAFF
```

5. An invalid use of parameter markers returns

EDA14041 USING clause has fewer values than ? in statement

which maps to SQLCODE=-313

Invalid use of ?.

Db2 returns the same SQLCODE.
An example of a statement that generates this type of error message is:

```
SELECT * FROM EDASERVE.X.QSTAFF
WHERE NAME = ?
```

6. A repeated GROUP keyword in the SELECT statement returns SQLCODE=-104, whereas Db2 returns SQLCODE=-199. An example of a statement that generates this type of error message is:

```
SELECT MANAGER
FROM EDASERVE.X.QORG
GROUP GROUP BY DIVISION
```

7. A join of greater than five server tables consumes a large amount of CPU power. This is a consequence of any Db2 application with a request of this nature.

8. An SQL statement greater than 256 columns in the ORDER BY clause returns the EDALOG message

```
E300010 THE NUMBER OF SORT FIELDS EXCEEDS THE MAXIMUM
```

and maps to SQLCODE=-136. In Db2 this statement returns an SQLCODE=-136

```
Sort key length is greater than 4000 bytes
```

when the string following the ORDER BY is greater than 4000 bytes.

An example of an SQL statement that generates this error is:

```
SELECT *
FROM EDASERVE.X.VPARTINV
ORDER BY name, name, name   (repeated 4000 times)
```

9. The NAME column in SYSCOLUM is 66 chars; in the Db2 catalog, this NAME column in SYSIBM.SYSCOLUMNS is 18 characters. When using prompted query in QMF to DESCRIBE the NAME column from SYSCOLUMNS, a view is created where the CNAME column is truncated from 66 chars to 30 chars, which is the maximum in the SQLDA. To create this view, append the following:

```
-INCLUDE EDALONG
-INCLUDE EDAQMFV
```

at the end of the user profile on the Server.

10. QMF's QBE DRAW of two tables with a WHERE clause generates a query where #$@OUTER is a correlation tag. The QBE generated query containing #$@OUTER as a correlation tag is presented to the server as a quoted string and the proper response is returned. An example of a QBE-generated SQL statement is:

```
SELECT #$@OUTER."NAME", #$@OUTER."ADDRESS"
FROM "EDASERVE"."X"."EMPLOYEE" #$OUTER
WHERE #$OUTER."NAME" = 'JAMES'
```
11. Using a built-in function in the WHERE clause, such as WHERE MIN(COLUMN) = 2, returns SQLCODE=-120

   A WHERE clause includes a column function

   which is what Db2 expects.

12. Incompatible columns in the UNION SELECT lists returns SQLCODE=-415

   Corresponding columns of the operands of a UNION do not have comparable column descriptions

   which is what Db2 returns as well.

   The following example returns SQLCODE=415

   ```sql
   SELECT NAME
   FROM EDASERVE.X.QSTAFF
   UNION
   SELECT EMPNO
   FROM EDASERVE.X.QSTAFF
   ```

13. Order of precedence using the NOT function in the WHERE clause, such as a statement containing WHERE NOT (A=B AND C=D) OR E=F, is performed according to the ANSI Level 2 standard.

14. QMF Prompted Query generates SELECT SALARY - 7.3E75 which returns in EDALOG

   EDA0202 INTERRUPT. FLOATING VALUE OVERFLOW

   which maps to SQLCODE=-802

   Exception error.

   Db2 returns SQLCODE=-405

   There is currently no specific parser detection of a floating value underflow.

15. In QMF Prompt Query LIST ? TABLE attempts to query a view DSQEC_TABS_RDB2. A member in EDARPC.DATA, named EDAQMFV, creates this view. To run this CREATE VIEW, place the following at the end of the user profile on the server, and have the FOCEXEC DD card defined with EDARPC.DATA to the server.

   ```
   -INCLUDE EDALONG
   -INCLUDE EDAQMFV
   ```
SAA CPI Functionality Checklist

The IBM Systems Application Architecture (SAA) Common Programming Interface (CPI) Functionality Checklist table, which follows the list of notes, evaluates Db2, the Extender for Db2, and server API support for the Extender for Db2. This table is based on the CPI checklist in the *SAA CPI Database Level 2 Reference Manual* (SC26-4798-00).

These notes pertain to the following table.

- Db2 support is for Version 11 running under z/OS.
- The Extender for Db2 refers to the SQL and associated statements that can be used to access server–controlled tables. The Extender for Db2 supports all Db2 functions for direct access to Db2.
- The server API refers to the SQL and associated statements that use server Translation Services. The Direct Passthru mode enables use of any SQL statement that can be accepted by the DBMS.
- *equivalent* signifies that the server API provides the equivalent function, but not through the same syntax.
- *future* signifies that this functionality is implemented in a future release, but has not been assigned to a particular release level yet.

<table>
<thead>
<tr>
<th>SQL Feature</th>
<th>Db2</th>
<th>Extender for Db2</th>
<th>Server API</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SELECT Expressions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT list</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FROM clause</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>WHERE clause</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>GROUP BY clause</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>HAVING clause</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ORDER BY clause</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNION</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNION ALL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Data Definitions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
</tr>
<tr>
<td>COMMENT ON</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
</tr>
<tr>
<td>CREATE INDEX</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
</tr>
<tr>
<td>CREATE TABLE</td>
<td>yes</td>
<td>n/a</td>
<td>yes</td>
</tr>
<tr>
<td>CREATE VIEW</td>
<td>yes</td>
<td>yes (1)</td>
<td>yes (1)</td>
</tr>
<tr>
<td>DROP</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
</tr>
</tbody>
</table>
## SQL Feature Checklist

<table>
<thead>
<tr>
<th>SQL Feature</th>
<th>Db2</th>
<th>Extender for Db2</th>
<th>Server API</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authorizations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRANT</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
</tr>
<tr>
<td>REVOKE</td>
<td>yes</td>
<td>n/a</td>
<td>no</td>
</tr>
<tr>
<td><strong>Basic Statements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Searched DELETE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>INSERT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SELECT INTO</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Searched UPDATE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>Cursor Operations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>DECLARE CURSOR</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>Positioned DELETE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>FETCH</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>OPEN</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>SELECT ... FOR UPDATE</td>
<td>yes</td>
<td>yes</td>
<td>future</td>
</tr>
<tr>
<td>Positioned UPDATE</td>
<td>yes</td>
<td>yes</td>
<td>future</td>
</tr>
<tr>
<td><strong>Dynamic Facilities</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIBE</td>
<td>yes</td>
<td>yes</td>
<td>yes (2)</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>yes</td>
<td>yes (3)</td>
<td>yes</td>
</tr>
<tr>
<td>EXECUTE IMMEDIATE</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>PREPARE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Connection and Transaction</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONNECT</td>
<td>yes</td>
<td>yes</td>
<td>yes (4)</td>
</tr>
<tr>
<td>COMMIT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Miscellaneous Statements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEGIN DECLARE SECTION</td>
<td>yes</td>
<td>yes</td>
<td>n/a</td>
</tr>
<tr>
<td>END DECLARE SECTION</td>
<td>yes</td>
<td>yes</td>
<td>n/a</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>yes</td>
<td>yes</td>
<td>n/a</td>
</tr>
<tr>
<td>LOCK TABLE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>OPTIMIZE FOR</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>WHenever</td>
<td>yes</td>
<td>no</td>
<td>n/a</td>
</tr>
<tr>
<td>SQL Feature</td>
<td>Db2</td>
<td>Extender for Db2</td>
<td>Server API</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
<td>-----------------</td>
<td>------------</td>
</tr>
<tr>
<td><strong>Data Types</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHARACTER</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DATE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FLOAT (single)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FLOAT (double)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>INTEGER</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TIME</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VARCHARAPhIC</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Column Functions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>COUNT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MAX</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MIN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SUM</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Scalar Function</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DATE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DAY</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DAYS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FLOAT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>HOUR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>INTEGER</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>LENGTH</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MICROSECOND</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MINUTE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MONTH</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SECOND</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SUBSTR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TIME</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VALUE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>YEAR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
### SQL Feature

<table>
<thead>
<tr>
<th>Feature</th>
<th>Db2</th>
<th>Extender for Db2</th>
<th>Server API</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicates</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALL/ANY</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>BETWEEN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>EXISTS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>IN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>IS NULL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>LIKE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Other Language Elements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arithmetic Operators</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Column References</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Comparison Operators</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Date/Time Arithmetic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Delimited Identifiers</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Foreign Keys</td>
<td>yes</td>
<td>no</td>
<td>no (5)</td>
</tr>
<tr>
<td>Host Variable</td>
<td>yes</td>
<td>no (5)</td>
<td>no (5)</td>
</tr>
<tr>
<td>Host Variable References</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>Indicator Variables</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Null Values</td>
<td>yes</td>
<td>no (6)</td>
<td>no (6)</td>
</tr>
<tr>
<td>Null Value Arithmetic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Ordinary Identifiers</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Primary Keys</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Search Conditions</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Special Registers</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>SQLCA</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>SQLDA</td>
<td>yes</td>
<td>yes</td>
<td>equivalent</td>
</tr>
<tr>
<td>SQLSTATE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>String Concatenation</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Host Languages</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Generator</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>C</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>COBOL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>PL/I</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Procedure Languages</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPG</td>
<td>yes</td>
<td>future</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Note:**

1. Views in the server system are temporary objects that exist only for the duration of your connection with that server. A view cannot be erased until the client session is terminated.

2. Also, enables the user to DESCRIBE a table without having to PREPARE a SELECT statement.
3. Not all SQL statements can be EXECUTEd. Only a previously PREPAREd SQL statement can be executed.

4. The server API enables a user to be connected to multiple servers simultaneously.

5. Host variable references within a SELECT are not supported. Parameter markers are supported.

6. A null value is treated as a zero in an arithmetic computation. For example, ten plus a null value results in a ten.

Extender for Db2 Cross-reference Table

Extender for Db2 users might receive SQLCODEs in the SQLCA that differ from expected SQLCODEs for similarly constructed SQL statements. To help diagnose the SQL request, the following table cross-references SQLCODEs received via the Extender for Db2 with possible Db2 SQLCODEs returned for equivalent Db2 SQL statements.

The following table also contains examples of SQL statements that generate the Extender for Db2 SQLCODE.

<table>
<thead>
<tr>
<th>Extender for Db2 SQLCODE</th>
<th>Possible Equivalent Db2 SQLCODE</th>
<th>Type of SQL Statement</th>
<th>See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>-84</td>
<td>-401</td>
<td>SELECT (USER+1).</td>
<td>Data Conversion on page 66</td>
</tr>
<tr>
<td>Extender for Db2 SQLCODE</td>
<td>Possible Equivalent Db2 SQLCODE</td>
<td>Type of SQL Statement</td>
<td>See ...</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------</td>
<td>-----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>-104</td>
<td>-115</td>
<td>Invalid use of &quot;&gt;&quot; as in WHERE SAL &gt; ANY NUMB.</td>
<td>Enhancements to Db2 on page 71</td>
</tr>
<tr>
<td>-171</td>
<td></td>
<td>Invalid syntax with VARGRAPHIC scalar function.</td>
<td>Data Conversion on page 66</td>
</tr>
<tr>
<td>-199</td>
<td></td>
<td>Repeated GROUP keyword in SELECT statement.</td>
<td>Enhancements to Db2 on page 71</td>
</tr>
<tr>
<td>-312</td>
<td></td>
<td>Invalid LIKE expression in WHERE clause.</td>
<td>Data Conversion on page 66</td>
</tr>
<tr>
<td>-904</td>
<td></td>
<td>SQL reserved words used as table names.</td>
<td>Data Conversion on page 66</td>
</tr>
<tr>
<td>Data Results</td>
<td>A function has an invalid expression.</td>
<td></td>
<td>Data Conversion on page 66</td>
</tr>
<tr>
<td>-802</td>
<td>-405</td>
<td>SELECT SALARY-7.3E75.</td>
<td>Data Conversion on page 66</td>
</tr>
<tr>
<td>-840</td>
<td>Data Results</td>
<td>Number of columns exceed the server limit of 256.</td>
<td>Enhancements to Db2 on page 71</td>
</tr>
<tr>
<td>-129</td>
<td></td>
<td>Db2 parses the FROM list and gets a limit exceeded; whereas the server parses the SELECT list limit first.</td>
<td>Enhancements to Db2 on page 71</td>
</tr>
<tr>
<td>Data Results</td>
<td>-107</td>
<td>Creator name in table reference is &gt; 8 characters.</td>
<td>Enhancements to Db2 on page 71</td>
</tr>
<tr>
<td>Data Results</td>
<td>-419</td>
<td>A server enhancement performs divide operation which results in a negative scale.</td>
<td>Enhancements to Db2 on page 71</td>
</tr>
<tr>
<td>Extender for Db2 SQLCODE</td>
<td>Possible Equivalent Db2 SQLCODE</td>
<td>Type of SQL Statement</td>
<td>See ...</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------</td>
<td>------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>-206</td>
<td>Not Permitted</td>
<td>Db2 does not permit tables with hyphenated column names.</td>
<td>General Considerations on page 63</td>
</tr>
<tr>
<td>-901ERRNUM</td>
<td>-105, -110, -115,-130, -159, -164,-207, -208, -414,-537, -553, -554,-603, -614, -815</td>
<td>The server does not support Db2-specific features.</td>
<td></td>
</tr>
</tbody>
</table>
This section provides examples of using specific third-party applications with the Extender for Db2.

In this chapter:

- Using QMF
- Using the Rocket Compiler for QMF
- Using COBOL

Using QMF

The example in this section illustrates how QMF can access VSAM data using the Extender for Db2, by:

- Presenting the run-time CLIST.
- Displaying an actual QMF query that accesses a VSAM table.
- Displaying the final screen of VSAM data returned to QMF.

Run-time CLIST

This sample run-time CLIST sets the run-time environment for QMF and the Extender for Db2. This CLIST is provided at qualif.HOME.DATA(EXTCQMF).
/* ------------------------- REXX -------------------------------*/
* Purpose: Sample CLIST to run QMF with the Extender             *
*                                                                *
* Pass the following parameters at invocation:                   *
*      qualif     High level qualifier for DB2 Extender datasets. *
*      db2hlq     High level qualifier for DB2 Libraries.          *
*      qmfhlq     High level qualifier for QMF Libraries.          *
*      dbss       DB2 Subsystem name.                             *
*----------------------------------------------------------------*/
parse upper arg qualif db2hlq qmfhlq dbss
"ALLOC FI(DSQLLIB) DA('qualif'.HOMEEXT.LOAD'," ,
  '"qualif'.HOME.LOAD'," ,
  '"ISP.SISPLLOAD'," ,
  '"qmfhlq'.SDSQLOAD'," ,
  '"db2hlq'.SDSNEXIT'," ,
  '"db2hlq'.SDSNLOAD') SHR REUSE"
"ALLOC FI(ADMCDATA) DA('GDDM.SADMCDA')      SHR REUSE"
"ALLOC FI(ADMDEFS)  DA('GDDM.ADMDEFS')      SHR REUSE"
"ALLOC FI(ADMGDF)   DA('GDDM.SADMGDF')      SHR REUSE"
"ALLOC FI(ADMSYMBL) DA('GDDM.SADSYM')       SHR REUSE"
"ALLOC FI(ADMCFORM) DA('"qmfhlq".SDSQCHRT') SHR REUSE"
"ALLOC FI(ADMGMAP)  DA('"qmfhlq".SDSQMAPE') SHR REUSE"
"ALLOC FI(DSQPNLE) DA('"qmfhlq".DSQPNL')   SHR REUSE"
"ALLOC FI(DSQEDIT) NEW UNIT(SYSALLDA) CYL SPACE(1 1)
  DSORG(FS) RECFM(F B A) LRECL(79) BLKSIZE(4029)"
"ALLOC FI(DSQDEBUG) DA(*)                  SHR REUSE"
"ALLOC FI(DSQPRINT) DA(*)                  SHR REUSE"
"ALLOC F(EDADPDS) DUMMY SHR"

/* copy EDAENV contents from sample EXTJQMF jcl */
"ALLOC F(EDAENV)   DA('qualif'.EDAENV')   SHR REUSE"

/* copy EDAENV contents from sample EXTJQMF jcl */
"ALLOC F(EDAENV)   DA('qualif'.EDAENV')   SHR REUSE"

/* copy EDACS3 contents from sample EXTJQMF jcl */
"ALLOC F(EDACS3)   DA('qualif'.EDACS3')   SHR REUSE"

/* ALLOC F(EDAPARMS) DA('qualif'.EDAPARMS') SHR REUSE */
/* copy IBITRACE contents from sample EXTJQMF jcl */
"ALLOC F(IBITRACE) DA('"qualif".IBITRACE') SHR REUSE"
"ALLOC F(FSTRACE) SYSOUT(X) RECFM(F) LRECL(132) BLKSIZE(132)"

"ALTLIB ACT APPL(CLIST) DA('"qmfhlq".SDSQCLTE')"
"ALTLIB ACT APPL(EXEC) DA('"qmfhlq".SDSQEXEC')"
address ispexec "LIBDEF ISPPLIB LIBRARY ID(DSQLLIB) STACK"
address ispexec "LIBDEF ISPMLIB DATASET ID('"qmfhlq".SDSQMLBE') STACK"
address ispexec "LIBDEF ISPSLIB DATASET ID('"qmfhlq".SDSQPLBE') STACK"
address ispexec "LIBDEF ISPSSLIB DATASET ID('"qmfhlq".SDSQSLBE') STACK"

/* QMF invocation */
address ispexec "SELECT PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB NOCHECK
  SCRNAME(QMF) PARM(DSQSSUBS="dbss")"
The SQL SELECT Statement

This screen displays a simple SQL SELECT statement. This is a query that a user would typically enter in QMF.

```
SQL QUERY                                            LINE 1

SELECT * FROM EDASERVE.X.VPARTINV

*** END ***
```

1=Help   2=Run    3=End     4=Print     5=Chart       6=Draw
7=Backward  8=Forward 9=Form  10=Insert   11=Delete   12=Report
OK, QUERY is displayed.

COMMAND ===>
SCROLL ===> PAGE

The Data Returned to QMF

This sample QMF screen shows the requested data returned to the client.
Using the Rocket Compiler for QMF

This section describes how to install the Extender for Db2 with the Rocket Software Compiler for QMF. For more information, see Installing the Extender for Db2 on z/OS on page 23.

Prerequisites

Confirm that your system has sufficient DASD to accommodate both the Rocket Compiler and the Extender for Db2 software. Also, if installing a server, see the appropriate server manual for specific hardware requirements. For more information on DASD memory requirements for the Extender for Db2, see Installing the Extender for Db2 on z/OS on page 23.

Installing the Rocket Compiler for QMF

Install the Rocket Compiler for QMF according to the documentation for Rocket QMF supplied by Rocket Software. For the Rocket Compiler, skip the step which creates a Db2 catalog snapshot of VSAM files. After the Rocket Compiler is installed, use the facility to generate, compile, and link-edit a COBOL program to verify independent functionality.
Installing the Extender for Db2

Users should already have QMF installed and running with Db2 on z/OS with the appropriate release levels. Also, verify that all of the client and server components are installed and fully functional. For more information, see the appropriate documentation.

To install the Extender for Db2 client, follow these steps, described in detail in Installing the Extender for Db2 on z/OS on page 23.

1. Allocate disk space for the Extender for Db2 libraries.
2. Unload the distribution tape.
3. Link-edit the main Extender for Db2 module with your Db2 entry points.
   
   **Note:** Db2 Extender is LE compliant and, therefore, requires all 3GL programs like COBOL to be linked using 31 bit addressing.

   `AMODE(31) RMODE(ANY)` for 31-bit addressing

   The Extender for Db2 main module is linked with 31 bit addressing mode in `qualif.HOME.LOAD`.

   Also, verify that the addressing mode is compatible in the generated Rocket Compiler JCL to compile and link-edit the generated COBOL2 program.

4. Set up the parameter file EDAPARMS (Optional). This step is optional and is dependent on site preferences and needs.
5. Link the Extender for Db2 statically to the Rocket-generated COBOL2 program. Use a modified version of Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional) on page 31.

   The Rocket Compiler for QMF generates and compiles QMF report programs in COBOL2, then performs a static link-edit to these programs. To ensure that the generated COBOL2 program can access the Extender for Db2, modify the link-edit step of the Rocket-generated JCL that generates, compiles, and links the Rocket-generated COBOL2 program. Place the Extender for Db2 `qualif.HOMEEXT.LOAD` and `qualif.HOME.LOAD` libraries in SYSLIB of the link-edit step, ahead of the standard Db2 load library, in concatenation sequence:

   ```
   //SYSLIB DD DISP=SHR, DSN=qualif.HOMEEXT.LOAD
   //      DD DISP=SHR, DSN=qualif.HOME.LOAD
   //      DD DISP=SHR, DSN=DSN810.SDSNLOAD
   ```

   Preserve all other libraries. Submit the Rocket-generated JCL. Your COBOL2 program is linked to the Extender for Db2 interceptor modules. After the generated COBOL2 programs are compiled and link-edited, they can be run according to the documentation for Rocket QMF.
6. Allocate the Extender for Db2 dynamically via Call Attach during Rocket Compiler user interface invocation.

Allocate the Extender for Db2 dynamically via Call Attach during Rocket Compiler user interface invocation. The Rocket Compiler for QMF generates QMF report programs in COBOL2. Users can run these generated programs from a panel, or they can invoke program execution by issuing a DSN RUN command from TSO. Both methods require dynamic allocation of the ROCKET.QMF.LOAD library in STEPLIB, or ISPLLIB in the interactive TSO environment. Therefore, in order to enable the Rocket Compiler for QMF in a QMF/Db2 environment, users must set up a run-time allocation CLIST that allocates the ROCKET.QMF.LOAD library ahead in the concatenation sequence in STEPLIB, or ISPLLIB.

Also, the Rocket Compiler invokes the standard Db2 load library via a dynamic Call-Attach load. Therefore, in order to enable both the Rocket Compiler and the Extender for Db2 for QMF in a QMF/Db2 environment, you must set up STEPLIB, or ISPLLIB, of your run-time CLIST or JCL (logon proc) by placing the Extender for Db2 qualif.HOMEEXT.LOAD and qualif.HOME.LOAD load libraries before your Db2 load library. An example of a JCL STEPLIB allocation is:

```
//STEPLIB DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//        DD DISP=SHR,DSN=qualif.HOME.LOAD
//        DD DISP=SHR,DSN=ROCKET.QMF.LOAD
//        DD DISP=SHR,DSN=QMF810.SDSQLOAD
//        DD DISP=SHR,DSN=DSN810.SDSNEXIT
//        DD DISP=SHR,DSN=DSN810.SDSNLOAD
```

After these changes, standard Rocket users who dynamically allocate the Db2 standard load libraries via Call Attach dynamically allocate the Extender for Db2 qualif.HOMEEXT.LOAD and qualif.HOME.LOAD libraries ahead of the Db2 load library.

A CLIST version of the allocation can also be used. For an example of a run-time CLIST, see *Installing the Extender for Db2 on z/OS* on page 23. Make modifications as needed.

You can change the concatenation order of the load libraries in STEPLIB, as long as you follow these basic allocation conditions.

- For enabling the Extender for Db2, allocate the Extender for Db2 load libraries before the standard Db2 load library.

- For enabling the Rocket Compiler, allocate the Rocket load library before the standard QMF load library.

7. Verify the operation of the Extender for Db2 independent of the Rocket Compiler for QMF before enabling the two products.
Using COBOL

These examples illustrate how to set up a COBOL2 application that sends requests for data to a server through the Extender for Db2. The examples are:

- Sample COBOL2 Program on page 89
- Sample Link-Edit JCL on page 104

Sample COBOL2 Program

The sample COBOL2 program, XTDCOB, performs the following:

- Processes both PREPARED and EXECUTE IMMEDIATE statements.
- Accesses both local Db2 tables and server (Extender) tables.
- Accepts commands from SYSIN DD.
- Accepts parameters from XTDPRM DD.

XTDCOB can be run in the following modes:

- Batch JCL mode or interactively.
- DSNALI mode or DSNELI mode.

You can enter the following commands in SYSIN DD for a Prepared SQL statement or an Execute Immediate statement.

```markdown
+------------------------------------------+
|            Available Commands:           |
|                                          |
| Prepared SQL statement Syntax:           |
|  SQL <sql statement>                     |
|  END                                     |
|                                          |
| Execute Immediate Syntax:                |
|  IMM <sql statement>                     |
|  END                                     |
|                                          |
| Type EXIT to exit this program.          |
+-----------------------------------------+
```

XTDCOB contains DEBUG and BATCH options for XTDPRM DD.

DEBUG=Y|N
where:

**Y**

Displays COBOL debugging messages.

**N**

Does not display the debugging messages. This is the default value.

**BATCH=Y|N**

where:

**Y**

Echoes input commands back to SYSOUT DD.

**N**

Does not echo input commands (interactive mode). This is the default value.

**Note:** The sample program can retrieve numeric and alpha columns, but it does not convert numeric columns into a displayable format. Therefore, the output of a numeric column appears in its internal binary representation. For example, the number 12 could be represented as 000C (undisplayable).
A sample XTDCOB program is provided at qualif.HOME.DATA(XTDCOB) follows:

IDENTIFICATION DIVISION.
PROGRAM-ID. XTDCOB.
AUTHOR. YANKO CASELLA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT XTDPRM
    ASSIGN TO XTDPRM.
SELECT DBGOUT
    ASSIGN TO DBGOUT.
DATA DIVISION.
FILE SECTION.
FD XTDPRM
  RECORD CONTAINS 80 CHARACTERS
  BLOCK CONTAINS 0 RECORDS
  LABEL RECORDS ARE OMITTED
  RECORDING MODE IS F.
  01 PRMREC
    PIC X(80).
FD DBGOUT
  RECORD CONTAINS 132 CHARACTERS
  BLOCK CONTAINS 0 RECORDS
  LABEL RECORDS ARE OMITTED
  RECORDING MODE IS F.
  01 MSGREC
    PIC X(132).
WORKING-STORAGE SECTION.
  01 MSGERR
    PIC X(132).
  77 MSGNUM
    PIC -ZZZ,ZZZ,ZZ9.
  77 MSGBLK
    PIC X(132) VALUE SPACES.
  77 MSGLEN
    PIC 9(4) VALUE 132.
  01 SLCT-STMT.
    49 SLCT-LENGTH
      PIC S9(04) COMP-5.
    49 SLCT-STRING
      PIC X(32700).
  01 SYSIN-STRING.
    05 SYSIN-CMD
      PIC X(04).
    05 SYSIN-LINE
      PIC X(76).
01 REXX-PARMS.
  05 FILLER    PIC X(6) VALUE "DEBUG=".
  05 DEBUG-YES PIC X.
  05 FILLER    PIC X(7) VALUE ",BATCH=".
  05 BATCH-YES PIC X.
  05 FILLER    PIC X(65).

***********************************************************************
* STRUCTURE FOR INPUT
***********************************************************************
01 IOAREA.
  02 TNAME    PIC X(72).
  02 FILLER   PIC X(08).

***********************************************************************
* VARIABLES FOR ERROR-MESSAGE FORMATTING
***********************************************************************
01 ERROR-MESSAGE.
  02 ERROR-LEN    PIC S9(4) COMP VALUE +960.
  02 ERROR-TEXT   PIC X(120) OCCURS 8 TIMES
                  INDEXED BY ERROR-INDEX.
  77 ERROR-TEXT-LEN PIC S9(8) COMP VALUE +120.

***********************************************************************
*    SQLDA
***********************************************************************
01 SQLDA.
  02 SQLDAID    PIC X(08) VALUE "SQLDA   ".
  02 SQLDABC    PIC S9(08) COMP VALUE 33016.
  02 SQLN       PIC S9(04) COMP VALUE 750.
  02 SQLD       PIC S9(04) COMP VALUE 0.
  02 SQLVAR     OCCURS 1 TO 750 TIMES
                DEPENDING ON SQLN.
                03 SQLTYPE    PIC S9(04) COMP.
                03 SQLLEN     PIC S9(04) COMP.
                03 SQLDATA    POINTER.
                03 SQLIND     POINTER.
                03 SQLNAME.
                49 SQLNAMEL   PIC S9(04) COMP.
                49 SQLNAMEC   PIC X(30).
  77 VARCTYPE    PIC S9(4) COMP VALUE +448.
  77 CHARTYPE    PIC S9(4) COMP VALUE +452.
  77 VARLTYPE    PIC S9(4) COMP VALUE +456.
  77 VARGTYPE    PIC S9(4) COMP VALUE +464.
  77 GTYPE       PIC S9(4) COMP VALUE +468.
  77 LVARGTYP    PIC S9(4) COMP VALUE +472.
  77 FLOATYPE    PIC S9(4) COMP VALUE +480.
  77 DECTYPE     PIC S9(4) COMP VALUE +484.
  77 INNTYPE     PIC S9(4) COMP VALUE +496.
  77 HWTYPE      PIC S9(4) COMP VALUE +500.
  77 DATETYP     PIC S9(4) COMP VALUE +384.
  77 MDTTIMTP    PIC S9(4) COMP VALUE +397.
EXEC SQL
  DECLARE SLCT-CSR CURSOR FOR SLCT-CSR-STMT
END-EXEC.

EXEC SQL
  DECLARE SLCT-CSR-STMT STATEMENT
END-EXEC.

EXEC SQL INCLUDE SQLCA END-EXEC.

77 ONE                  PIC S9(4) COMP VALUE +1.   01380099
77 TWO                  PIC S9(4) COMP VALUE +2.   01390099
77 FOUR                 PIC S9(4) COMP VALUE +4.   01400099
77 QMARK                PIC X          VALUE "?".   01401099
77 LAST-CMD             PIC X(4).                01410099
* SQL RETURN CODE HANDLING                          *
*****************************************************
EXEC SQL WHENEVER SQLERROR  GOTO DBERROR END-EXEC.
EXEC SQL WHENEVER NOT FOUND  CONTINUE    END-EXEC.
OPEN INPUT XTDPRM.
READ XTDPRM INTO REXX-PARMS.
CLOSE XTDPRM.
IF DEBUG-YES = 'Y' THEN
  OPEN OUTPUT DBGOUT.
  DISPLAY "+---------------------------------+".
  DISPLAY "|       Available Commands:       |
  DISPLAY "| Prepared SQL statement Syntax: |
  DISPLAY "|  SQL <sql statement>            |
  DISPLAY "|  END                            |
  DISPLAY "| Execute Immediate Syntax:      |
  DISPLAY "|  IMM <sql statement>           |
  DISPLAY "|  END                            |
  DISPLAY "| Type EXIT to exit this program.|
  DISPLAY "+---------------------------------+".
  MOVE ONE TO SLCT-LENGTH.
MAIN-LOOP.
  DISPLAY "Type Command To Be Processed:".
  DISPLAY " ".
  MOVE SPACES TO LAST-CMD.
READ-SQL.
  ACCEPT  SYSIN-STRING FROM SYSIN.
  IF BATCH-YES = 'Y' THEN
    DISPLAY SYSIN-STRING.
EVALUATE FUNCTION UPPER-CASE (SYSIN-CMD)
WHEN "SQL "
  MOVE "SQL " TO LAST-CMD
  MOVE ONE TO SLCT-LENGTH
  STRING SYSIN-LINE DELIMITED BY SIZE
    INTO SLCT-STRING WITH POINTER SLCT-LENGTH
  GO TO READ-SQL

WHEN "IMM "
  MOVE "IMM " TO LAST-CMD
  MOVE ONE TO SLCT-LENGTH
  STRING SYSIN-LINE DELIMITED BY SIZE
    INTO SLCT-STRING WITH POINTER SLCT-LENGTH
  GO TO READ-SQL

WHEN "EXIT"
  GO TO PROG-END

WHEN "END"
  SUBTRACT ONE FROM SLCT-LENGTH

IF DEBUG-YES = 'Y' THEN
  STRING "LAST-CMD = ", LAST-CMD, MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
  WRITE MSGREC FROM MSGERR
  MOVE SLCT-LENGTH TO MSGNUM
  STRING "IN READ-SQL SLCT-LENGTH = ", MSGNUM, MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
  WRITE MSGREC FROM MSGERR
  STRING "IN READ-SQL SLCT-STRING = ", SLCT-STRING,
    MSGBLK DELIMITED BY MSGLEN INTO MSGERR
  WRITE MSGREC FROM MSGERR
END-IF

EVALUATE LAST-CMD
WHEN "SQL"
  PERFORM PROCESS-INPUT THROUGH IND-RESULT
WHEN "IMM"
  PERFORM IMMED-SQL THROUGH PRINT-ROWS
END-EVALUATE
GO TO MAIN-LOOP

WHEN OTHER
  STRING SYSIN-STRING DELIMITED BY SIZE
    INTO SLCT-STRING WITH POINTER SLCT-LENGTH
  GO TO READ-SQL
END-EVALUATE.
IF DEBUG-YES = 'Y' THEN
CLOSE DBGOUT.
GOBACK.

* PREPARE

PROCESS-INPUT.
EXEC SQL
PREPARE SLCT-CSR-STMT INTO :SQLDA FROM :SLCT-STMT
END-EXEC.

* SET UP ADDRESSES IN THE SQLDA FOR DATA

IF DEBUG-YES = 'Y' THEN
MOVE SQLD TO MSGNUM
STRING "IN PROCESS-INPUT SQLD = ", MSGNUM, MSGBLK
DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR
MOVE SQLN TO MSGNUM
STRING "IN PROCESS-INPUT SQLN = ", MSGNUM, MSGBLK
DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.

* IF STATEMENT IS NOT SELECT, EXECUTE STMT

IF SQLD = ZERO THEN
GO TO NOT-A-SELECT.

DISPLAY " "
MOVE ZERO TO ROWCOUNT.
MOVE ZERO TO REC1-LEN.
SET RECPTR TO ADDRESS OF REC1-CHAR(1).
MOVE ONE TO I.
INITIALIZE TITLE-LINE.
MOVE ONE TO TITLE-LEN.
PERFORM COLADDR UNTIL I > SQLD.
MOVE SPACES TO SQLDATA-REC(5:REC1-LEN).
DISPLAY TITLE-LINE.
DISPLAY TITLE-SEP.
** ******************************************************************** 02760099
  *           SET LENGTH OF OUTPUT RECORD.                  * 02770099
  *           OPEN CURSOR                                   * 02780099
** ******************************************************************** 02790099

EXEC SQL OPEN SLCT-CSR END-EXEC.

** ******************************************************************** 02800099
  * FETCH                                                  * 02810099
** ******************************************************************** 02820099

IF DEBUG-YES = 'Y' THEN
STRING "AT FETCH.....", MSGBLK DELIMITED BY MSGLEN
    INTO MSGERR
    WRITE MSGREC FROM MSGERR.

EXEC SQL
    FETCH SLCT-CSR
    USING DESCRIPTOR :SQLDA
END-EXEC.

IF SQLCODE = ZERO THEN
PERFORM WRITE-AND-FETCH
    UNTIL SQLCODE IS NOT EQUAL TO ZERO.

MOVE ROWCOUNT TO MSGNUM
DISPLAY " 
DISPLAY "***** NUMBER OF RECORDS IN TABLE=" MSGNUM " *****"
DISPLAY " 
IF DEBUG-YES = 'Y' THEN
STRING "LEAVING FETCH.....", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    WRITE MSGREC FROM MSGERR.
CLOSEDT.
IF DEBUG-YES = 'Y' THEN
STRING "AT CLOSEDT.....", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    WRITE MSGREC FROM MSGERR.

EXEC SQL CLOSE SLCT-CSR END-EXEC.

IND-RESULT.
IF DEBUG-YES = 'Y' THEN
STRING "AT IND-RESULT. RETURNING TO MAIN LOOP.", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    WRITE MSGREC FROM MSGERR.

7. Using Extender for Db2 Application Implementations
WRITE-AND-FETCH.
  IF DEBUG-YES = 'Y' THEN
  STRING "AT WRITE-AND-FETCH.....", MSGBLK
  DELIMITED BY MSGLEN INTO MSGERR
  WRITE MSGREC FROM MSGERR.
  MOVE ONE TO INDCOUNT.
  PERFORM NULLCHK UNTIL INDCOUNT > SQLD.
  IF DEBUG-YES = 'Y' THEN
    STRING "SQLDATA-REC = ", SQLDATA-REC, MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    WRITE MSGREC FROM MSGERR.
    DISPLAY SQLDATA-REC(5:REC1-LEN).
    MOVE SPACES TO SQLDATA-REC(5:REC1-LEN).
    ADD ONE TO ROWCOUNT.
    EXEC SQL
      FETCH SLCT-CSR
      USING DESCRIPTOR :SQLDA
    END-EXEC.
  IF DEBUG-YES = 'Y' THEN
    STRING "IN WRITE-AND-FETCH SQLDA = ", SQLDA, MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    WRITE MSGREC FROM MSGERR.
    STRING "LEAVING WRITE-AND-FETCH.....", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    ITE MSGREC FROM MSGERR.
    NULLCHK.
    IF DEBUG-YES = 'Y' THEN
      STRING "AT NULLCHK.....", MSGBLK
      DELIMITED BY MSGLEN INTO MSGERR
      WRITE MSGREC FROM MSGERR
      MOVE IND(INDCOUNT) TO MSGNUM
      STRING "IN NULLCHK IND(INDCOUNT) = ", MSGNUM, MSGBLK
      DELIMITED BY MSGLEN INTO MSGERR
      WRITE MSGREC FROM MSGERR.
      IF IND(INDCOUNT) < 0 THEN
        SET ADDRESS OF SQLDATA-BLANK TO WORKINDPTR(INDCOUNT)
        MOVE QMARK TO INDREC.
    IF DEBUG-YES = 'Y' THEN
MOVE INDCOUNT TO MSGNUM
STRING "IN NULLCHK AFTER IF - INDCOUNT = ", MSGNUM, MSGBLK
   DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.

ADD ONE TO INDCOUNT.

IF DEBUG-YES = 'Y' THEN
   STRING "LEAVING NULLCHK....", MSGBLK
      DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.

COLADDR.
   IF DEBUG-YES = 'Y' THEN
      STRING "AT COLADDR.....", MSGBLK
         DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.

SET SQLDATA(I) TO RECPTR.
****************************************************************
*             DETERMINE LENGTH OF COLUMN (COLUMN-LEN)          *
****************************************************************
MOVE SQLLEN(I) TO COLUMN-LEN.
****************************************************************
*             COLUMN-IND IS 0 FOR NO NULLS AND 1 FOR NULLS     *
****************************************************************
DIVIDE SQLTYPE(I) BY TWO GIVING DUMMY REMAINDER COLUMN-IND.
****************************************************************
*             MYTYPE IS JUST THE SQLTYPE WITHOUT THE NULL BIT  *
****************************************************************
MOVE SQLTYPE(I) TO MYTYPE.

IF DEBUG-YES = 'Y' THEN
   MOVE SQLTYPE(I) TO MSGNUM
   STRING "IN COLADDR SQLTYPE(I) = ", MSGNUM, MSGBLK
      DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.

SUBTRACT COLUMN-IND FROM MYTYPE.
**SET THE COLUMN LENGTH, DEPENDENT UPON DATA TYPE**

EVALUATE MYTYPE
    WHEN CHARTYPE CONTINUE,
         ADD TWO TO COLUMN-LEN,
    WHEN DATETYP THROUGH MDTTIMTP CONTINUE,
    WHEN FLOATYPE CONTINUE,
    WHEN VARCTYPE
         ADD TWO TO COLUMN-LEN,
    WHEN VARLTYPE
         ADD TWO TO COLUMN-LEN,
    WHEN GTYPE
         MULTIPLY COLUMN-LEN BY TWO GIVING COLUMN-LEN,
    WHEN VARGTYPE
         PERFORM CALC-VARG-LEN,
    WHEN LVARGTYP
         PERFORM CALC-VARG-LEN,
    WHEN HWTYPE
         MOVE TWO TO COLUMN-LEN,
    WHEN INTTYPE
         MOVE FOUR TO COLUMN-LEN,
    WHEN DECTYPE
         PERFORM CALC-DECIMAL-LEN,
    WHEN OTHER
         PERFORM UNRECOGNIZED-ERROR,
END-EVALUATE.

IF DEBUG-YES = 'Y' THEN
    MOVE COLUMN-LEN TO MSGNUM
    STRING "IN COLADDR COLUMN-LEN = ", MSGNUM, MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR
    WRITE MSGREC FROM MSGERR.

    ADD COLUMN-LEN TO RECNUM.
    ADD COLUMN-LEN TO REC1-LEN.

    STRING SQLNAMEC(I) DELIMITED BY SPACE
        INTO TITLE-LINE POINTER TITLE-LEN.
    STRING SPACE DELIMITED BY SIZE
        INTO TITLE-LINE POINTER TITLE-LEN.
*IF THIS COLUMN CAN BE NULL, AN INDICATOR VARIABLE IS NEEDED*

**PROCEDURE FOR NULL INDICATOR**

1. Move zero to IND(I).
2. If COLUMN-IND = ONE, set SQLIND(I) to the address of IND(I) and set WORKINDPTR(I) to RECPTR.
3. Add one to RECNUM and REC1-LEN.
4. Add one to I.

**PROCEDURE FOR DECIMAL COLUMN LENGTH**

1. If DEBUG-YES = 'Y', string "LEAVING COLADDR....", MSGBLK delimited by MSGLEN into MSGERR.
2. Write MSGREC from MSGERR.
3. Calculate COLUMN-PREC by dividing COLUMN-LEN by 256 and giving the remainder COLUMN-SCALE.
4. Move COLUMN-PREC to COLUMN-LEN.
5. Add one to COLUMN-LEN.
6. Divide COLUMN-LEN by two, giving COLUMN-LEN.
7. If DEBUG-YES = 'Y', move COLUMN-LEN to MSGNUM.
8. String "IN CALC-DECIMAL-LEN COLUMN-LEN= ", MSGNUM, MSGBLK delimited by MSGLEN into MSGERR.
9. Write MSGREC from MSGERR.
10. String "LEAVING CALC-DECIMAL-LEN....", MSGBLK delimited by MSGLEN into MSGERR.
11. Write MSGREC from MSGERR.
*PERFORM PARAGRAPH TO CALCULATE COLUMN LENGTH FOR A VARGRAPHIC DATA TYPE COLUMN.
CALC-VARG-LEN.
  IF DEBUG-YES = 'Y' THEN
    STRING "AT CALC-VARG-LEN.....", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR.
    WRITE MSGREC FROM MSGERR.
  MULTIPLY COLUMN-LEN BY TWO GIVING COLUMN-LEN.
  ADD TWO TO COLUMN-LEN.
  IF DEBUG-YES = 'Y' THEN
    STRING "LEAVING CALC-VARG-LEN.....", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR.
    WRITE MSGREC FROM MSGERR.
*PERFORM PARAGRAPH TO NOTE AN UNRECOGNIZED DATA TYPE COLUMN.
UNRECOGNIZED-ERROR.
  MOVE MYTYPE TO MSGNUM
  DISPLAY "UNRECOGNIZED DATA TYPE = " MSGNUM
  MOVE COLUMN-LEN TO MSGNUM
  DISPLAY " COLUMN-LEN = " MSGNUM
  GO TO IND-RESULT.
NOT-A-SELECT.
  IF DEBUG-YES = 'Y' THEN
    STRING "AT NOT-A-SELECT....", MSGBLK
    DELIMITED BY MSGLEN INTO MSGERR.
    WRITE MSGREC FROM MSGERR.
  EXEC SQL
    EXECUTE SLCT-CSR-STMT USING DESCRIPTOR :SQLDA
  END-EXEC.
  PERFORM PRINT-ROWS.
  GO TO IND-RESULT.
IMMED-SQL.
IF DEBUG-YES = 'Y' THEN
STRING "AT IMMED-SQL....", MSGBLK
DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.
EXEC SQL
EXECUTE IMMEDIATE :SLCT-STMT
END-EXEC.
PRINT-ROWS.
MOVE SQLERRD(3) TO MSGNUM
DISPLAY " "
DISPLAY "***** NUMBER OF ROWS AFFECTED BY REQUEST=" MSGNUM
" *****"
DISPLAY " 
IF DEBUG-YES = 'Y' THEN
STRING "LEAVING PRINT-ROWS.....", MSGBLK
DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.
DBERROR.
IF DEBUG-YES = 'Y' THEN
STRING "AT DBERROR.....", MSGBLK
DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR
MOVE SQLCODE TO MSGNUM
STRING "SQL ERROR OCCURRED, SQLCODE = ", MSGNUM,
MSGBLK DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR.
CALL "DSNTIAR" USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN.
IF RETURN-CODE = ZERO
PERFORM ERROR-PRINT VARYING ERROR-INDEX FROM 1 BY 1 UNTIL ERROR-INDEX GREATER THAN 8
ELSE
IF DEBUG-YES = 'Y' THEN
STRING "DSNT4971 RETURN CODE FROM MSG ROUTINE DSNTIAR",
MSGBLK DELIMITED BY MSGLEN INTO MSGERR
WRITE MSGREC FROM MSGERR
END-IF.
ELSE.
ENDIF.
ENDIF.
ENDIF.
ELSE
STRING "LEAVING DBERROR....", MSGBLK
DELIMITED BY MSGLEN INTO MSGERR.
WRITE MSGREC FROM MSGERR.
ENDIF.
IF LAST-CMD = "SQL " THEN
GO TO IND-RESULT
ELSE
GO TO PRINT-ROWS.
ENDIF.
ERROR-PRINT.
IF DEBUG-YES = 'Y' THEN
STRING "AT ERROR-PRINT....", MSGBLK
DELIMITED BY MSGLEN INTO MSGERR.
WRITE MSGREC FROM MSGERR.
DISPLAY ERROR-TEXT (ERROR-INDEX).
IF DEBUG-YES = 'Y' THEN
STRING "LEAVING ERROR-PRINT....", MSGBLK
DELIMITED BY MSGLEN INTO MSGERR.
WRITE MSGREC FROM MSGERR.

Sample Link-Edit JCL

This JCL sample, XTDCLG, precompiles, compiles, link-edits and executes COBOL2 programs containing dynamic SQL with the Extender for Db2. XTDCLG does the following.

- Precompiles, compiles, and links program XTDCOB. Program XTDCOB accepts commands from SYSIN DD, either in batch or interactively (CLIST). It can be run in either DSNALI or DSNELI mode.
The syntax for SYSIN DD is:

```
+------------------------------------------+
| Available Commands:                     |
|                                          |
| Prepared SQL statement Syntax:           |
| SQL <sql statement>                     |
| END                                      |
|                                          |
| Execute Immediate Syntax:                |
| IMM <sql statement>                     |
| END                                      |
|                                          |
| Type EXIT to exit this program.          |
+------------------------------------------+
```

- Executes XTDCOB.

To link-edit your JCL, perform the following steps.

1. Change all instances of `qualif`, `user`, `db2hlq`, `dbss`, `hostn` and `portn` to match your site specifications. See comments in the jcl at `qualif.HOME.DATA(XTDCLG)`.  
2. Copy COBOL source member XTDCOB from `qualif.HOME.DATA` into your COBOL source library. Make the necessary changes to the SYSIN DD card in the PC step.  
3. Confirm that you have followed the installation instructions for the Extender for Db2 as described in Chapter 2, *Installing the Extender for Db2 on z/OS*. Verify that you have link-edited the Extender for Db2 main module with your Db2 entry points, if your site has a local Db2 subsystem. If you have properly link-edited the Extender for Db2 main module with your Db2 entry points, the `qualif.HOMEEXT.LOAD` library referenced in STEPLIB of the RUNSTEP in the XTDCLG JCL should be properly linked to your site’s local Db2 subsystem.  
4. Submit the JCL. The output of the SQL request is found in SYSOUT.

**Note:** If an application only processes servers (no local Db2 access), it is not necessary to bind and grant the customer application plan. The server default plan (dynamic plan) is used instead.
XTDCLG

/**
 *        Job Card Goes Here
 /**
 /**
 /** Note: DSNELI could be used instead, DSNALI was used arbitrarily.
 /**
 /**
 /**Substitutions:
 /**  qualif - High level qualifier for DB2 Extender datasets
 /**  db2hlq - High level qualifier for DB2 libraries.
 /**  user   - High level qualifier for user libraries.
 /**  dbss   - DB2 Subsytem name.
 /**  hostn  - Server's Host name or Server's IP address.
 /**  portn  - TCP/IP Port number server is listening on.
 tokenId
 **/ //************************************************************
 //         SET DB2REL=db2hlq
 //************************************************************
 /**
 PC XDTCOB
 //************************************************************
 //PC EXEC PGM=DSNHPC,
 //STEPLIB DD DISP=SHR,DSN=&DB2REL..SDSNEXIT
 // DD DISP=SHR,DSN=&DB2REL..SDSNLOAD
 //SYSIN DD DISP=SHR,DSN=user.COBOL.SOURCE(XTDCOB)
 //DBRMLIB DD DISP=SHR,DSN=user.DBRMLIB.DATA(XTDCOB)
 //SYSCIN DD DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
 //         SPACE=(800,(500,500))
 //SYSPRINT DD SYSOUT=* 
 //SYSTERM DD SYSOUT=* 
 //SYSUT1 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
 //SYSUT2 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
 //SYSUT3 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
 //SYSUT4 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
 //SYSUT5 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//****************************************************
//*               BIND STEP FOR XTDCOB
//* (only required to use DB2 directly (NATIVELY))
//****************************************************
//BIND      EXEC   PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB   DD DSN=&DB2REL..SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//SYSTSPRT DD SYSOUT=* 
//DBRMLIB  DD DISP=SHR,DSN=user.DBRMLIB.DATA(XTDCOB)
//SYSTSIN   DD *
//DSN SYSTEM(dbss)
BIND PLAN      (XTDCOB)   - 
    MEMBER    (XTDCOB)   - 
    LIBRARY   ('user.DBRMLIB.DATA') - 
    ACTION    (REPLACE) - 
    ISOLATION (CS) - 
    ACQUIRE   (USE) - 
    RELEASE   (COMMIT) - 
    EXPLAIN   (YES) 
END 
/*
//****************************************************
//*               GRANT STEP FOR XTDCOB
//* (only required to use DB2 directly (NATIVELY))
//****************************************************
//GRANT     EXEC   PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB   DD DSN=&DB2REL..SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN   DD *
RUN PROGRAM (DSNTIAD) PLAN (DSNTIA81) - 
    LIB ('DSN810.RUNLIB.LOAD') 
END 
/*
//SYSSIN DD * 
GRANT EXECUTE ON PLAN XTDCOB TO PUBLIC; 
/*
RUN STEP for XTDCOB

//RUNXTD EXEC PGM=XTDCOB,COND=((4,LT),EVEN)
//STEPLIB DD DISP=SHR,DSN=user.COBOL.LOAD
// DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
// DD DISP=SHR,DSN=qualif.HOME.LOAD
// DD DISP=SHR,DSN=&DB2REL..SDSNEXIT
// DD DISP=SHR,DSN=&DB2REL..SDSNLOAD
//EDACS3 DD *
NAME = Client Odin File

NODE = EDASERVE
BEGIN
  PROTOCOL = TCP
  CLASS = CLIENT
  HOST = hostn ;Server's Host name or IP address
  PORT = portn ;Port # server is listening on
  TRACE = 31
END

//EDAENV DD *
FSTRACE=DD:FSTRACE
EDACONF=/PDS
/*
//EDADPDS DD DUMMY
//*EDAPARMS DD DISP=SHR,DSN=user.EDAPARMS
//IBITRACE DD *
SET TRACEON=ALL
/*
//FSTRACE DD SYSOUT=*,DCB=(LRECL=132,RECFM=FB,BLKSIZE=132)
//SYSOUT DD SYSOUT=* 
//DBGOUT DD SYSOUT=* 
//XTDPRM DD *
DEBUG=N,BATCH=Y 
/*
//SYSIN DD *

SQL
SELECT COUNTRY,CAR,MODEL,BODYTYPE FROM EDASERVE.ANYNAME.CAR
END
SQL
SELECT LAST_NAME,FIRST_NAME FROM EDASERVE.ANYNAME.EMPLOYEE
END
EXIT
/*
This section lists server and Extender for Db2 messages and codes. It also provides a cross-reference to Db2 SQLCODEs.

**In this appendix:**
- API Status Codes
- Server Error Codes and SQLCODEs
- Extender for Db2 Error Codes

### API Status Codes

API status codes are converted to SQLCODEs and SQLERRMC (which reside in SQLCA). The following chart shows some of the API status codes and conversions.

<table>
<thead>
<tr>
<th>API Status Code</th>
<th>SQLCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>+100</td>
</tr>
<tr>
<td>-9</td>
<td>-904 for Db2</td>
</tr>
<tr>
<td>-12</td>
<td>-904 for Db2</td>
</tr>
<tr>
<td>Most other negative server status codes</td>
<td>Default ERRNUM (if not defined, then -901)</td>
</tr>
</tbody>
</table>

### Server Error Codes and SQLCODEs

The following table illustrates some of the current conversions of server codes to SQLCODEs.

<table>
<thead>
<tr>
<th>Server Code</th>
<th>SQLCODE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDA0000</td>
<td>0</td>
<td>Status OK</td>
</tr>
</tbody>
</table>
## Server Error Codes and SQLCODEs

<table>
<thead>
<tr>
<th>Server Code</th>
<th>SQLCODE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDA0003</td>
<td>-206</td>
<td>Field not found</td>
</tr>
<tr>
<td>EDA0005</td>
<td>-840</td>
<td>Too many columns</td>
</tr>
<tr>
<td>EDA0010</td>
<td>-136</td>
<td>Too many sort keys</td>
</tr>
<tr>
<td>EDA0016</td>
<td>-206</td>
<td>Field not found</td>
</tr>
<tr>
<td>EDA0201</td>
<td>-802</td>
<td>Division by zero</td>
</tr>
<tr>
<td>EDA0202</td>
<td>-802</td>
<td>Floating point overflow</td>
</tr>
<tr>
<td>EDA0203</td>
<td>0</td>
<td>Floating point underflow</td>
</tr>
<tr>
<td>EDA0205</td>
<td>-204</td>
<td>File not found</td>
</tr>
<tr>
<td>EDA0236</td>
<td>-206</td>
<td>Field not found</td>
</tr>
<tr>
<td>EDA0258</td>
<td>-206</td>
<td>Field not found</td>
</tr>
<tr>
<td>EDA0277</td>
<td>-414</td>
<td>Invalid format in LIKE column</td>
</tr>
<tr>
<td>EDA0281</td>
<td>-401</td>
<td>Incompatible types</td>
</tr>
<tr>
<td>EDA0370</td>
<td>-206</td>
<td>Field not found</td>
</tr>
<tr>
<td>EDA0486</td>
<td>0</td>
<td>File allocated</td>
</tr>
<tr>
<td>EDA0582</td>
<td>0</td>
<td>SQL syntax error</td>
</tr>
<tr>
<td>EDA0757</td>
<td>0</td>
<td>Multi-path DB-ignore</td>
</tr>
<tr>
<td>EDA4204</td>
<td>-204</td>
<td>No logical PCB for file</td>
</tr>
<tr>
<td>EDA4211</td>
<td>-204</td>
<td>No BMP region has PCB</td>
</tr>
<tr>
<td>EDA4902</td>
<td>0</td>
<td>Model 204 database restarted; informational</td>
</tr>
<tr>
<td>EDA14007</td>
<td>-104</td>
<td>Syntax error</td>
</tr>
<tr>
<td>EDA14009</td>
<td>-204</td>
<td>File not found</td>
</tr>
<tr>
<td>EDA14010</td>
<td>-206</td>
<td>Field not found</td>
</tr>
<tr>
<td>Server Code</td>
<td>SQLCODE</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>EDA14012</td>
<td>-203</td>
<td>Column name ambiguous</td>
</tr>
<tr>
<td>EDA14013</td>
<td>-84</td>
<td>Unsupported SQL syntax</td>
</tr>
<tr>
<td>EDA14014</td>
<td>-401</td>
<td>Incompatible operands</td>
</tr>
<tr>
<td>EDA14015</td>
<td>-412</td>
<td>Too many columns on subquery</td>
</tr>
<tr>
<td>EDA14018</td>
<td>-132</td>
<td>Invalid pattern on LIKE</td>
</tr>
<tr>
<td>EDA14025</td>
<td>-421</td>
<td>Incompatible UNION</td>
</tr>
<tr>
<td>EDA14026</td>
<td>-415</td>
<td>Incompatible UNION</td>
</tr>
<tr>
<td>EDA14028</td>
<td>-125</td>
<td>Invalid ORDER BY number</td>
</tr>
<tr>
<td>EDA14030</td>
<td>-601</td>
<td>Table exists on CREATE</td>
</tr>
<tr>
<td>EDA14041</td>
<td>-313</td>
<td>Too many parameter markers</td>
</tr>
<tr>
<td>EDA14043</td>
<td>-612</td>
<td>Duplicate column name</td>
</tr>
<tr>
<td>EDA14045</td>
<td>-601</td>
<td>View exists on CREATE</td>
</tr>
<tr>
<td>EDA14046</td>
<td>-204</td>
<td>View not found</td>
</tr>
<tr>
<td>EDA14053</td>
<td>-122</td>
<td>Invalid GROUP BY clause</td>
</tr>
<tr>
<td>EDA14056</td>
<td>-122</td>
<td>Invalid GROUP BY clause</td>
</tr>
<tr>
<td>EDA14058</td>
<td>-127</td>
<td>Invalid use of DISTINCT</td>
</tr>
<tr>
<td>EDA14064</td>
<td>-112</td>
<td>Invalid aggregate function</td>
</tr>
<tr>
<td>EDA14065</td>
<td>-120</td>
<td>Invalid aggregate function</td>
</tr>
<tr>
<td>EDA14066</td>
<td>-129</td>
<td>Too many tables in SQL</td>
</tr>
</tbody>
</table>

**Extender for Db2 Error Codes**

The Extender can generate its own error codes into a specific SQLCODE and SQLERRMC (in SQLCA). The following are some of the Extender-generated Db2 codes.
<table>
<thead>
<tr>
<th>Server Event</th>
<th>Db2 Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>An overflow is detected.</td>
<td>+802</td>
</tr>
<tr>
<td>An unacceptable SQL statement is found.</td>
<td>-084</td>
</tr>
<tr>
<td>Too many tables or views are in a request.</td>
<td>-129</td>
</tr>
<tr>
<td>A null value cannot be assigned to output host variable because no indicator variable is specified.</td>
<td>-305</td>
</tr>
<tr>
<td>The number of parameter markers does not match the number of SQLDA entries.</td>
<td>-313</td>
</tr>
<tr>
<td>Unsupported use of parameter markers.</td>
<td>-418</td>
</tr>
<tr>
<td>A cursor in a FETCH or CLOSE statement is not OPEN.</td>
<td>-501</td>
</tr>
<tr>
<td>A cursor in an OPEN statement is already OPEN.</td>
<td>-502</td>
</tr>
<tr>
<td>An SQL request references multiple locations.</td>
<td>-512</td>
</tr>
<tr>
<td>A DESCRIBE was performed on an UNPREPAREd statement.</td>
<td>-516</td>
</tr>
<tr>
<td>A PREPARE statement identifies the SELECT statement of the OPENed cursor.</td>
<td>-519</td>
</tr>
<tr>
<td>A user ID does not have privilege to perform operation or the server is Read/ Only.</td>
<td>-551</td>
</tr>
<tr>
<td>An attempt is made to CONNECT when an application is not in a CONNECTable state.</td>
<td>-752</td>
</tr>
<tr>
<td>An overflow is detected.</td>
<td>-802</td>
</tr>
<tr>
<td>The main module QXQMFX has not been linked to IBM DSNALI, DSNHLI, DSNTIAR.</td>
<td>-901</td>
</tr>
<tr>
<td>With Db2, an unsuccessful execution is caused by an unavailable resource. Most likely, the server is not connected.</td>
<td>-904</td>
</tr>
<tr>
<td>With SQL/DS, an unsuccessful execution is caused by an unavailable resource. Most likely, the server is not connected.</td>
<td>-940</td>
</tr>
</tbody>
</table>
Connecting to Multiple Servers

The Extender for Db2 supports the use of SQL CONNECT to connect to multiple servers or to retrieve information about the currently connected server. Use the command to connect to a server that you specify or to the server specified in the EDAPARMS file.

The Db2 connection states of connectable and connected/unconnected, and of unconnectable and connected are similar to the Db2 design and fully supported. For more information, see the IBM Db2 Reference Manual. Use SQL CONNECT functionality in the same manner as in a multiple Db2 subsystem environment.

In this appendix:

- Explicitly Connecting to a Server
- Implicitly Connecting to a Server
- Retrieving Information About a Server
- Error Messages

Explicitly Connecting to a Server

Use the following form of SQL CONNECT to establish the current server for the client application process. The current server must be a valid server or a valid Db2 subsystem.

```
EXEC SQL CONNECT TO :host variable
EXEC SQL CONNECT TO location
```

where:

- `host variable` is a host variable.

- `location` is an explicit location. For the explicit designation of the current server, the location name must be a valid server or a valid Db2 subsystem.

- To be a valid server, its definition must reside under the communications configuration file.
For a valid Db2 subsystem, the Extender for Db2 does not recognize any local or remote Db2 subsystems, but passes the CONNECT request to Db2, for Db2 to make the validation. In this manner, remote Db2 subsystems can be connected to the Extender for Db2, provided that the local Db2 subsystem, which the Extender for Db2 was originally link-edited at installation can recognize the remote Db2 subsystem name.

Upon a COMMIT or ROLLBACK, the Extender for Db2 resolves and dispatches partially-qualified table requests to the current server.

Use the explicit form of SQL CONNECT whenever partially-qualified tables must be resolved to a Server for Db2 or to a server.

Implicitly Connecting to a Server

Use the following form of SQL CONNECT to establish the server defined in the EDAPARMS file as the current server:

EXEC SQL CONNECT RESET

Issuing this command is equivalent to issuing

EXEC SQL CONNECT TO default server

where:

default server

Is the server defined in the EDAPARMS file.

Upon a COMMIT or ROLLBACK, the Extender for Db2 resolves and dispatches partially-qualified table requests to the current server.

The default server is defined in the EDAPARMS file under the EDASERVE keyword. If the local Db2 subsystem is the desired default, then omit the EDASERVE assignment in the EDAPARMS file.

Retrieving Information About a Server

Use the following form of SQL CONNECT to return information about the current server in the SQLERRP field of the SQLCA.

EXEC SQL CONNECT

For the Extender for Db2, the information returned in the SQLERRP field is:

AKB02000

The connected server or location may be explicitly or implicitly defined.
**Error Messages**

An unsuccessful SQL CONNECT TO request returns a SQLCODE of -752, and an SQLSTATE of 51011 in the SQLCA.
Index

A

adapter-specific password security 52
allocation streams 33
  login environment and; allocation
  streams: batch address space and; CAF
  explicit load of DSNALI module; explicit load of DSNALI module; 33
alternate column names and SQL translation 58
ANSI level 2 60
answer sets 58, 70
API/SQL
  status codes 111

C

Call-Attach Facility (CAF) 16, 24, 25, 33, 38
cartesian product, and SQL translation services 58
CHAR data type conversion 67
client security 53
COBOL, and Extender for Db2 89
column name resolution, and SQL translation 58
communications configuration (EDACS3) file 51, 56
converting data 67
copying entry points 29
creating a user library 29

D

data conversion 67
data sets
  unloading from tape 28
Db2 attachment facilities 24
Db2 entry points
  linking to main Extender module; 30
Db2 explicit loading 25

Db2 subsystem 25
answer set displays 70
Call-Attach Facility (CAF) 25, 33, 38
data conversion 67
DSN Command Processor 26, 40
enhancements with Extender for Db2 71
error handling 67, 70
Extender for Db2 16
IBM SAA CPI checklist 75
interceptor modules 24, 30, 32, 33
non-ANSI compliant SQL requests 69
parameter marker support 68
SQLCODE cross-reference table 79
usage considerations 63

DBA password security 52
DECIMAL data type conversion 67
disk requirements for Extender for Db2 24
DSN Command Processor 16, 24, 26, 40
DSNALI module 16, 24, 26, 30, 32
DSNELI module 16, 32
DSNHLI module 25, 30
DSNHLI2 module 30
DSNTIAR module 16, 32
  z/OS installation 25, 30, 32
  z/OS tracing 54
dynamically defined virtual fields and SQL translation 58

E
EDA/SQL server 51, 56
  fully-qualified tables 56
  partially-qualified tables 56
EDAACLG sample program 104
EDACOB2 sample program 91
EDAENV trace file 54
EDAPARMS file 15, 45, 115
  partially-qualified tables 56
  product work tables 57
  security in z/OS 51
  z/OS installation 31
EDASERVER keyword in EDAPARMS file 49
ENABLE keyword in EDAPARMS file 48
entry points 30
ERRNUM keyword in EDAPARMS file 46, 49
Error codes 113
error handling 67, 70
error message formatting facility (DSNTIAR) 54
error messages and codes 111
explicit loading of Db2 25
Extender for Db2 16
  answer set displays 70
  application implementations 83
  COBOL and 89
  Connecting to servers 115
  copying entry points 29
  data conversion 67
  Db2 subsystem usage considerations 63
  disk requirements 24
  EDAPARMS file 45
  enhancements to Db2 71
  Error codes 113
  error handling 67, 70
  features 15
  functional overview 16
  hardware requirements 23
  IBM SAA CPI checklist 75
  installing in z/OS without Db2 43
  non-ANSI compliant SQL requests 69
  operation overview 16
  parameter marker support 68
  QMF and 83
  Rocket Compiler for QMF 86
  security 51
  SQL considerations 55
  SQL translation 57
  SQLCODE cross-reference table 79
  tracing 53
INDEX

F
FLOAT data type conversion 67
FSTRACE trace file 53
fully-qualified tables 55

G
GENEQX member 30
GENEQXI member 32
general purpose SELECT statements 15

H
hardware requirements for Extender for Db2 23

I
IBITRACE trace file 54
IBM Call-Attach Facility (CAF) 15
IBM DSN Command Processor 15
IBM SAA CPI checklist 75
installation worksheet 27
installing Extender for Db2 27
installing Extender for Db2 in z/OS 25

L
link-editing DSNALI module 26
link-editing standard modules 38
link-editing with Db2 entry points 30
linking main Extender module with Db2 entry points 30

M
message formatting facility (DSNTIAR) 16, 54
parameter marker support 68
partially-qualified tables 55, 56
PREPARE statement 15
preparing run-time allocations 33
product work tables 55, 57

Q
QMF and Extender for Db2 83
Distributed Data Facility (DDF) 24
qualif.EDHOME.DATA 24
qualif.EXTHOME.DATA 24
QXUDUMP trace file 54
QXUDUMP trace files 54

Extender for Db2
R

Rocket Compiler for QMF and Extender for Db2 86
run-time allocations 33

S

security 51
  User ID/password 52
  z/OS 51
SELECT statements 15
server
  connecting to 45, 115
setting default error codes 15
setting parameters in EDAPARMS 31
SMALLINT data type conversion 67
software requirements for Extender for Db2 23
SQL CONNECT command 115
SQL connect verbs 15
15
SQL considerations 55
SQL functions 15
SQL general purpose statements 15
SQL join improvements 61
SQL Passthru 75
SQL SELECT statements 15
SQL table naming 55
SQL translation services 57, 75
  column name resolution 58
dynamically defined virtual fields 58
  limitations 61
performance enhancements 61
SQL join improvements 61
view creation/deletion 61
virtual column support 60
SQLCA format 20, 67
SQLCODEs 70, 79, 111
cross-reference table 79
SQLDA format 20, 66
SQLERRMC 111
system requirements for Extender for Db2 24

T

table naming conventions 16, 55
tracing 53
  z/OS 53

U

unloading data sets from tape 28
User ID/password security 52

V

VARCHAR data type conversion 67
view creation/deletion 61
virtual column support 60
W

WORKTABLE keyword in EDAPARMS file 48, 49, 56, 57

Z

z/OS and Extender for Db2 25
Call-Attach Facility (CAF) 25, 33, 38

z/OS and Extender for Db2 25
DSN Command Processor 26, 40
functional overview 16
installation without Db2 43
pre-installation 24
security 51
tracing 53
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.
Extender for Db2

Release 77 and Higher